The central goal in the development of Software Production Environments (SPEs) is to provide a framework for the integration of a set of tools providing support in the whole software development process. Under a high degree of integration the SPE provides capabilities for the interoperation of tools and the incremental and consistent development of all descriptions produced during the development process. A high degree of integration can be achieved with a common uniform user interface and with a common shared information repository that enables tools to share all information which is available at each stage of the development process. Two essential components of any SPE are, therefore, a common sophisticated window management system and object management system. However, the above mentioned concept of a SPE still does not support a more industrialized production of software that is ultimately aiming at a further automation of the whole software process to produce software in a way similar to other engineering processes like car manufacturing or plant construction.

First the great number of activities being performed by the tools of a SPE has to be structured and ordered according to a formal process model. In more traditional approaches a process model could just reflect a phase oriented procedure where sequential ordering of steps are being performed in order to produce the final product. Those phase oriented models are usually not appropriate because in most cases they do not reflect issues like the distributed development of software by a large number of (geographically dispersed) persons in possibly even different organizations. Furthermore, the software is usually not produced in a sequence of steps but different activities are highly interlocked. Therefore, the structuring of the activities is usually based on the definition of roles which persons involved in the software development are supposed to play. According to its role(s) a certain person performs or has to perform certain activities and has access to specific tools (e.g. [2]).

Second, one particular process model cannot be assumed for all the needs of all prospective users. In order to provide flexibility and easy adaptation of defined models they should be generic, i.e. by setting values of certain parameters it must be possible to adjust a generic process model to the needs of a special company or even single user.

This customization of generic process models is usually a complex process itself and should therefore be structured itself. In a first step, one can identify at least three different levels of customization. On the "company level" all the parameters specific to the needs of a particular company are fixed. On a "project level" the parameters defining the peculiarities of a certain project are fixed. Finally, on a "running project" level the process...
model must still be adjustable to a changing situation in the currently running project (cf. [2], [3]).

The more industrialized and efficient production of software is only possible when the concept of reuse of already existing software components is part of the process. Any process model should therefore reflect the storage of reusable components for later reuse as well as the search for a particular already existing component in order to reuse it within the currently developed software.

Process models providing support for the so far mentioned aspects are usually very complex descriptions. Adequate specification methods and languages are absolutely necessary to be able to precisely describe a particular (generic) process model. Such methods and languages based on different approaches have, therefore, been proposed in a number of projects. Examples are an entity-relationship oriented approach in the IPSE 2.5 project [4], a Petri-Net based approach in ALF [5] and PIMS [6], a language based on graph grammars in the IPSEN project [7], and a rule based language in ESF [2] or Marvel [8]. All these mentioned languages have only been applied in paper studies or in very first prototype implementations so far. So they all still lack an extensive use and consequently a proof of their applicability.

In most of the above mentioned approaches the complexity of process modelling, however, has already stimulated to exploit expert systems techniques in order to overcome some of the difficulties which arise due to the fuzzyness of the problem area itself.

The development of a SPE (or as it is nowadays often called a Software Factory) is in itself a first example of a large software project which itself requires a clearly defined process model. The ESF project (EUREKA Software Factory [9]) comprising of 13 European companies and research institutions, therefore apply a bootstrapping approach to software process modelling. By developing a process model and according methods and languages for the construction of ESF itself the process modelling area is extensively studied. Thereby, methods and languages for the definition and implementation of process models in general are being developed.