Chapter 3
Process Modelling Languages

Editors: Reidar Conradi and M. Letizia Jaccheri

3.1 Introduction

This chapter deals with the requirements and proposed solutions for process modelling languages (PMLs). Process modelling is a very diverse and complex area and the requirements for the support of modelling and execution are both technical (e.g. for expressiveness, abstraction, and multiple perspectives) as well as non-technical (e.g. for commercial support).

As mentioned in Chapter 1, a PML expresses software production processes in the form of a process model, being a computer-internal description of an external process. In the last 10 years, many PMLs have been proposed, implemented and experimented on without any real general agreement or standardization among them.

Nevertheless, general agreement is emerging (see Chapter 1, for instance) on identifying the primary process elements such as activities (tasks), products, roles, tools, agents, and evolution support. A first basic requirement is that these concepts have to be captured by the language. The first chapter has also introduced the notion of the meta-process (explored and described in more detail in Chapter 4) which provides a second basic requirement for the PML in that each meta-process phase, such as Elicitation, must be supported by a language that allows the model to be expressed at the appropriate abstraction level.

Each process element, i.e. tasks, roles, products and so on, needs to be described. Other elements playing a secondary role in the context of a process model are the notions of work contexts, tool views, and user views. Finally, cooperation models, versioning and transaction, and quality models have to be described. In this chapter we will distinguish between the primary and secondary elements.

Now a PML can be formal, semi-formal, or informal. A formal PML is one which is provided with a formal syntax and semantics. Semi-formal languages usually have a graphical notation with formal syntax, but not formal semantics i.e. not being executable. Natural languages, such as English, may be used as informal PMLs. In this chapter, the emphasis is on formal PMLs, as they provide support for formal verification and analysis, simulation (i.e. “dry runs”), and execution (“wet runs”).
The fundamental design challenges are thus:

1. Should we have a small kernel PML, and many specialised sub-PMLs for process sub-domains, or should we have one large PML, with sub-PMLs as views? And what about interoperability in federated systems, with sub-PMLs and related model repositories?

2. Should one PML be used to support each meta-process phase or should we have many PMLs, each devoted to a different meta-phase? Advantages and disadvantages stem from those in their respective domain. If the same language is used across different phases, then it easier to keep the different models consistent and to reuse knowledge across the different phases. On the other hand, it is evident that a single language is seldom sufficient to support the whole life cycle of a deliverable, from its conception to its implementation and operation.

A general software architecture for a PML comprises at least (see Chapter 5): a repository in which the model is stored and maintained in a versioned fashion; an interpreter to execute models; an editor to facilitate model creation and manipulation. The PML design choices influence the architectural choices, e.g., there can be several incompatible PMLs, where each must be provided with its own interpreter and editor, or there can be several compatible PMLs in the sense that they can be executed by the same interpreter. This will be discussed.

There are obviously results from current research and technology that have the potential to be adapted for the process modelling domain, and we will therefore look at related areas, such as information modelling, databases and groupware for ideas and suitable solutions.

This chapter is structured as follows: Section 3.2 presents some requirements for PMLs. This includes both the software process elements and the process modelling phases that have to be described by a PML. It discusses views connected to process roles and meta-process roles such as those of project managers, process engineers, process agents etc. It also deals with general modelling demands, such as understandability and modularization. Section 3.3 introduces the theoretical background for PMLs, by identifying the main conceptual sources and linguistic solutions for existing PMLs. Section 3.4 and Section 3.5 give a summary of some existing PMLs, and classify these with respect to linguistic paradigms and the nature of the support offered. Conclusions and directions for future research are offered in Section 3.7.

3.2 Requirements on Process Modelling Languages

A software process model is a representation of the real-world activities of a software production process. A software process model is developed, analysed, refined, transformed and/or enacted within the meta-process. Thus, any software process model has to model the real-world process appropriately and it has to meet the specific requirements of each phase of the meta-process. These imply requirements for the process modelling language by which a process model is defined. These requirements on the