An Example of Process Change

Robert Snowdon

ICL, West Avenue, Kidsgrove, Stoke on Trent, ST7 1TL, England

Abstract. Processes in businesses and other organisations are not set in concrete. They have to be flexible and responsive to stimuli which cause them to change. Process technology must reflect these characteristics if it is to find realistic application. This paper illustrates how the evolutionary nature of processes may be supported in a particular process technology, that originally developed in the IPSE 2.5 project.

1 Introduction

The traditional view of software development is that it is a process starting with the establishment of requirements, moving through phases such as design and implementation and finishing in something called “maintenance”. Of course, we all know that reality is rather different with the process being rather more cyclic in nature, phases overlapping one another and so on. Process technology must not be restricted to the simple view if it is to find credible application. Change and concurrency are essential elements of this and many other application domains.

Such concerns were taken into account in the thinking behind the IPSE 2.5 project [4] and in the process technology originating in that project and subsequently developed as the PS system and the language PML [1].

This paper illustrates the particular features of this technology which are concerned with these aspects. It does this by means of a concrete example utilising a particular paradigm of change developed in the IPSE 2.5 project. This paradigm is known as PMMS (Process Model for Management Support). It provides an integration of the notion that a process is something which is developed and changed as a process itself.

Example PML encodings are included, the intention being to allow the reader to see in detail what is going on, what needs to be described and how it is described. A brief explanation of the form and effect of PML descriptions is covered in sections 2 and 3. This should be sufficient for the purposes of this paper although it is, of course, not a complete definition of PML or PS.

2 Some Basic Points about PS and PML

PML is a language in which a process description is expressed. The primary purpose of a description is to be the basis for the enactment of the process by appropriate “agents” (people and tools). Enactment support is provided by means of a computer system (called

* This work was supported by a grant from the Royal Society and the Science and Engineering Research Council while the author was a Visiting Fellow at the Department of Computer Science, University of Manchester, England
the PCE, the Process Control Engine) which uses PML descriptions to provide appropriate capabilities to the agents participating in the process itself. More details of these concepts can be found in, for example, [4],[1] and [5].

In PML a process is considered as comprising a number of roles which communicate through interactions.

A role can be thought of as being rather like a single-threaded process having a private set of resources and some knowledge of its environment. It may help to envisage a role as being "played" by a person, although strictly this is rather a limited interpretation of a more general concept. An interaction is an asynchronous, uni-directional, buffered channel by which roles communicate.

PML is a language based on the idea of Class. The primitive notions of role and interaction are supported by Role Class and Interaction Class definitions respectively. A class defines properties for its members, objects which are instances of the class. PML also supports Entity Class and Action Class definitions of which such properties may be formed. Subclassing is supported (with some restrictions) through an isa hierarchy.

The predefined classes of PML are available in all roles. A role (that is an instance of Role Class) may also possess a set of class definitions of its own. A newly created role will possess class definitions depending on its definition and the class definitions of its creator.

A role exhibits behaviour. That is, actions operate on the resources of a role and possibly on its environment according to a description of behaviour associated with the role via its membership of a particular subclass of Role Class. This behaviour is described in terms of actions which may occur if appropriate conditions (known as triggers) are satisfied. The single threaded nature of roles means that only one action will occur at any one time.

PML provides a number of predefined subclasses of Action Class, though new subclasses may also be defined. Of particular note for this paper are definitions of action subclasses to create new roles (the subclass StartRole) and to modify existing roles (the subclass BehaveAs). Actions are provided to support communication between roles in the form of the subclasses Give and Take. Interactions are identified in roles by GivePorts and TakePorts, specialised as to the objects that may be transmitted.

Finally, PML incorporates the idea of Agents. An agent represents a tool or person participating in a process. The Agent Class is a subclass of Role Class, so agents are roles and communicate with other roles via interactions. In practice PML provides specially defined actions (of UserAction Class) which encapsulate much of the communication between roles and agents which represent people. The differences between agents which represent tools and agents which represent people are captured by predefined subclasses ToolAgent and UserAgent of Agent Class.

3 The Starting Position

Every newly delivered PS system is already primed to support the enactment of a very simple process. This has two roles, one of which is a user agent supporting the participation of a person known as 'root'. The other role is known as base and is defined to interact with the 'root' user agent in a very specific way as shown in the following PML extract.

---

2 PML constructs for tools are not discussed here.