Abstract
The last few years have afforded the author an opportunity to think ‘beautiful thoughts’ on Executable Process Modelling (XPM).

The first part of this paper sets out the thinking and principles behind the ProML language – a comprehensive language for recording process for execution. The second part of the paper addresses some of the perceived issues around XPM, its possible uses and the realities and consequences of its use.

It is assumed in this paper that readers are familiar with at least the principles of Executable (or Enactable) Process Modelling.

The Language
ProML (Process Modelling Language) was designed as an exercise in going back to first principles with as few presuppositions as possible but inevitably informed by distant memories of the IPSE 2.5 PML language. That language always appeared too object-purist – the ProML language was not to be academic but down-to-earth. The language was to be comprehensive and comprehensible, a vehicle for experimentation with methods of supporting human and machine activity electronically. It had to be universal, applicable to any activity and not just business or the technical design / implementation process.

The semantics were defined fairly closely but the syntax chosen in the first instance was textual although other syntaxes, perhaps graphical, could be created later for the existing semantics.

The fundamental elements of the language are Entity (the ‘done to’), Activity (the ‘doing’) and Role (the ‘doer’). The element Actor defines real-world entities, human or mechanical, that carry out Roles. Activities are governed by pre-, post- and co-conditions, Actions by logical constructs. Logical constructs and conditions are rules of arbitrary complexity that yield a logical outcome.

The language is not object-oriented – it relies on element template cloning for creating new instances of elements. It is not algorithmic but is, unless unavoidable, of keyword-with-parameters shape – it describes process, not algorithms in the traditional sense. There are no traditional value-carrying variables. It has no scoping at present – this is deemed an unnecessary complication. There are no iteration constructs as conditions (pre- and post-) govern repetition. It has no communication mechanisms between elements as the states of entities can mimic any communication paradigm simply. A simple form of fuzzy logic is available in its rule processing.

Constructs are included in the language to support particular aspects of activity, like diaries and plans.

The connection between process-modelling-world elements and real-world elements is defined by simple one-to-one / one-to-many equivalency constructs within element definitions. The real-world looks after the passing of information, the process model mediates the communication of signals.

Communication with the real-world user of the system would be by to-do lists, perhaps occasionally couched in terms of wizards, but it is felt that there must be a better way, as yet not obvious…
The metaphor for a process model implemented in ProML is the ‘Boiling Cauldron’: a big pot of process model elements soup running by themselves according to their definition in an open-ended way, accepting or creating additional elements at any time and, as a result of this, adapting its behaviour to the new soup mix. The cooking fire is the XPM interpreter engine. Elements of the process model need to be capable of self-reference and this is possible in the defined language to some extent but probably not enough. At present, elements can clone each other and also modify each other to a limited extent. This self-reference is essential to enable a process modelling system to reason about itself modify itself in order to adapt to new situations. It is not needed in many sort-lived applications but becomes essential if intelligently evolving process models to support complex and long-lived organisations are to be possible. Very little thought has been given to this until now by the author.

The implementation of this language would, by the nature of its role, support concurrency. Since it is envisaged that a ProML model could run and evolve for an extended period of time, the usual garbage-collection and recycling of resources would also have to be catered for. A client-server system would run the model on a process server and send its real-world communication and activities to clients using the usual networking technology.

Execution of ProML will be by interpretation to an intermediate language which will not consist of bytecodes but binary-relational database tuples. The Binary-Relational information model permits the definition of data to arbitrary levels of semantic subtlety. Execution of this intermediate language will be based on SQL querying of the intermediate code. The chosen intermediate code representation will make it easier to verify, report on and correct on the fly – this, of course, remains to be proved. In-core databases such as TimesTen™ could keep up, it is believed, with the pace required for keeping the cauldron boiling efficiently.

Other thoughts
Process Modelling is applicable to all activities people or their mechanical or electronic artefacts carry out. For example, there may be an application in supporting the lifestyle of people with cognitive dysfunctions, like Alzheimer’s. Unfortunately, the original ambition to have a language that was easily understood by ‘ordinary’ people has not yet been achieved. A facility to translate ProML into an English-like equivalent presentation could probably be devised. Although this would not have the formal rigour of the original, it could convey the meaning of process models in a way that is natural to human beings.

Reversibility of a process model’s execution to an arbitrary distance back in time under the process model’s control would be useful when circumstances, such as an internal failure or change in the context of the model warrants it. This will take some thinking about.

Some parallels between a process model ‘cauldron’ and living cells is apparent. The clonable template elements of the model are similar to genes. The actual cloning of elements from these templates parallels the building of proteins/enzymes from the genes. The process of cloning can be controlled in the language by precondition rules so that, as in living systems, the creation of a process element can be the result of changes in the process model and its environment, just as genes are switched on by the presence of metabolites that need to be dealt with. In other words, a process model could be thought of as having a ‘genome’, which is the collection of template elements and the parts of the model that control their cloning. This genome would
define the behaviour and adaptability of the model just as its genome defines a creature and its capabilities. All this opens the possibility of process memes, perhaps spread by process viruses and even process mating …

Lastly, if a process model system gets complex enough, it may tip into chaotic behaviour or, more interestingly, hover around the boundary of order and chaos, where living systems are placed. This border region is thought to be not only relatively stable but also capable of creative adaptation of its system to its changing environment.

**Acknowledgements**

The author want to thank several student interns who have helped him over the years in thinking about XPM, tried it out and built some parts of an interpreter. In particular David Poole of UWE and Jim Farrand of the University of Bristol. Also members of the Informatics Process Group at the University of Manchester and System Modelling Research Group at the University of the West of England, for interesting discussions.

This work was carried out when a member of the Design Technology Group of Philips Semiconductors in Southampton although this paper does not represent the views or any commitment of that company. The original involvement with this field occurred when the author was the Plessey team Leader in the Alvey IPSE 2.5 project in the late eighties.

---

**Note…**

I do not expect this paper to be published so I have not gone to too much trouble in formatting it or making it particularly formal.