Reflections on the relationship between BPR and Software Process Modelling

Brian Warboys

Informatics Process Group (IPG)
Department of Computer Science
University of Manchester M13 9PL

Abstract. *Business Process Re-engineering* (BPR) is much in vogue both as an approach to the rationalisation of Corporate Organisations and as an aid to the design of the IT system which supports the Organisation.
The field of *Software Process Modelling* (SPM) started some 10 years ago with the objective of modelling and thence supporting the total set of software engineering activities necessary to develop and maintain software products.
They both arose as a result of dissatisfaction with the current approaches to their respective domains. They share the word *process* but is there any other connection between them? This paper attempts to outline some possible synergies and pitfalls which might derive from a closer contact between the two disciplines.

Keywords:*Software Process Modelling , Business Process Re-engineering*

1 Introduction

1.1 Some Definitions

Feiler and Humphrey [1] define a *Process* as a *A set of partially ordered steps intended to reach a goal.* Another definition from the Workflow community is *A process is characterised by a number of individuals acting to fulfil a set of (ideally) common goals.* The first gives us a scientific base the second a sociological one.

A *Process Model* is a description of a *Process* expressed in a suitable *Process Modelling Language* (possibly diagrammatic). A model is always an abstraction of the reality it represents, and as such, it is only a partial description.

The machine execution of a *Process Model* is termed *enaction.* The term is used to distinguish it both from the notion of computer program execution and from the notion of the real-world process execution which it supports. There has been much debate in the Software Process Modelling community about the relationship between conventional program execution and process enactment. This debate is not of concern to this paper but it is still convenient to refer to the accepted term.

1.2 On Process-Oriented Initiatives

The current environment is full of so called *Process-Oriented* initiatives. This phenomena can perhaps be explained by social and business conditions which have created a trend towards a *Globalisation Plus Individualisation* approach to business. On the one hand arguing that competitiveness is only obtained on a Global scale and on the other that Individuals should be *empowered* to act in ways that they think are individually best suited to meet the objectives which they have been set or set for themselves.

This is essentially free market economics of the most extreme kind. Our actions should have the greatest global impact whilst allowing any so called legal means to be exploited at an individual level. Entrepreneurial paradise!

In order for such a system to function there is a clear need to delineate personal responsibilty, not in terms of individual morality, but rather in terms of the legal and corporate processes which both constrain the individual and also define the means by which individuals cooperate with others. If we imagine a future where a large amount of trading, and work in general, is performed by individuals isolated at computer terminals then increasingly such *rules* are defined by the capabilities of the Information Systems which provides the basic means of interacting *individually* on a *Global* stage. Such capabilities include personal telephones, faxes, laptop computers, modems, electronic mail networks and Databases.

This has lead, and increasingly will continue to lead, on the one hand to the *re-engineering* of corporations to provide such personalised *empowered* environments and on the other hand the growth of *Workflow* technology to support collaborative office automation.

The origins of Software Process Modelling are somewhat different. The desire to control the *Software Process* is nearly as old as software itself. In 1968 and 1969 two NATO sponsored workshops [2] were held which arguably invented the term *Software Engineering*. The term was necessary as it was perceived that Software Systems had grown to such complexity that they needed an engineering method to guide their development. The last 25 years have led to all manner of methods being proposed. A quick glance at the ACM SIGSOFT Software Engineering Notes reveals the inadequacy of these approaches. Software continues to both kill people and disrupt social behaviour.

Some 10 years ago the focus for Software Engineering support started to move away from an essentially product-design centered approach towards a more process-centered approach. Hitherto Software Engineering Environments had been built around the notion of a database schema description of the product being developed, being used to *manage* a set of development tools. A similar approach to a conventional commercial TP system.

This change of focus has lead to work in Process-Centred Development Environments, essentially controlling co-ordination through an enacted description of the software process to be used to develop the product. This, in turn, has lead to the study of Software Process Modelling (SPM) as a discipline in its own right.

Given the current state of maturity (or rather lack of it) of the two fields of BPR and SPM it seems appropriate to reflect on the connections between the two topics; if only because they share the word *process*.

2 On Business Process Re-engineering (BPR)

The term *re-engineering* was arguably first applied by Mike Hammer [3] in the Harvard Business Review 1990, proclaiming "Re-engineering work: Don't automate, Obliterate!". Clearly an approach which seeks to minimise bureacracy rather than institutionalising it in computer programmes has some appeal. However I believe that the sentiment fails to deal with the essential evolutionary nature of ecosystems. In fact the general BPR approach could be said to suffer from a tendency to treat organisations as machines rather than ecosystems.

Apart from the huge investment in legacy computer systems, the stored knowledge of a corporation lies in the very imprecise processes which regulate individual's behaviour. Rather like Lewis Thomas' remark in "The Medusa and the Snail" - "The capacity to blunder slightly is the real marvel of DNA", most individuals "get the job done" adjusting process to meet the immediate demands.

Leaving aside this issue of "Re-engineering Corporations" the other key message from BPR is the one of concern to this particular paper.

Following on from work at MIT in the "Management in the 90's" programme there is now considerable focus on the need to integrate the Corporation's 'IT' Strategy with the Business Strategy. Numerous case-studies are quoted to show that there is considerable advantage in designing the IT system in terms of how it supports business processes. Further, that by such integration, one can modify trading practises to take full advantage of electronic trading to offer new and more profitable services.

This has had three major effects:

- Corporations must model their business processes before they can automate them. This has the excellent side-effect of gaining understanding of the business.
- It casts considerable doubt on the accepted practise of a centralised Corporate database ER model supporting a set of Transactions actioned through appropriate VDU screens. This happens to arrive at the same instance as the plea for "Open Systems" and the "down-sizing of Mainframes" in favour of UNIX servers. A happy co-incidence!
- It encourages a rapid investment in co-ordination technology, workflow etc, since in order to support business processes the technology needs to be able to cope with the co-ordination of workgroups.

So far so good. However the natural temptation is now to reason:

- the business is managed as a set of business processes
- these business processes are modelled in software
- the IT strategy should be integrated with the business strategy

– designing the software is *just* part of designing the business

Although clearly the design of the software does radically impact the design of the business it ignores two important points:

– The software system is still a software system with all the old problems of versioning, integration, complexity etc.
– The rules governing good architectural practises in software are not the same as the rules governing good business structures.

These problems are magnified when we take into account the fact that 'workflow' software is of the most complex type from a software engineering perspective. Workflow programmes are parallel, asynchronous and should be able to evolve as the business processes which they support evolve.

3 On Software Process Modelling (SPM)

Process Modelling, in general, derives from three seperate concerns:

– The Transaction Model is insufficient to describe modern trading patterns. There is a need for the notion of 'Very Long Transactions' which allow transactions to both last a long time (e.g the Life Insurance Policy Transaction lasts literally for a lifetime) and to cross corporate boundaries to allow for inter-enterprise trading.
– The growth of PC networks demands coordination software to allow intelligent mail and office workflow to be supported.
– In order to allow for the flexible integration of heterogeneous systems a loose framework architecture needs to be established. Essentially the role performed by a Shell script as the personal binding to a UNIX workstation needs to be performed for a workgroup connected by a network of PCs or Workstations.

This last requirement will, I believe, turn out to be the one of most significance. Suppose we imagine our future IT systems to be completely distributed with individual terminals specialised to support specific personal roles. Terminals would have, on the one hand, unlimited access to world-wide hypertext and multi-media information bases. And, on the other hand, very restricted and personalised access to Corporate Information bases.

How do we provide the infrastructure, the constraints and support for team working in this environment?

Given that the central repository has disappeared, and that the naming structure for the network is distributed then we cannot, any longer, control these systems from the centre. Thus they need to be controlled from the user-end, with a little help from the service providers.

This implies a 'Shell'-like programme which operates at the workstation but links each user to peer collaborators and to tools and information bases in terms

of the job to be performed rather than some organisational 'schema'. This is Process Model Enactment.

Software Process Modelling is a particular brand of process modelling concerned with support for the software process. It shares the same concerns as any other form of process modelling but with three extra challenges:

- The software process is very fine-grain.
- The software process is very *soft* with the need for much customerisation in order to deal both with the rapid change to the product being produced and to the methods used in that production.
- Because, the product to be built and the process model used to support this build are both implemented as software, it offers the opportunity for 'reflexive' systems. That is systems which contain the means of their own evolution.

It is because of these concerns, which have lead to more general approaches, that the technology developed in the field of software process modelling has been found to be immediately applicable to other domains. The converse is not true, workflow office automation products have not been able to be successfully applied to the support of software engineering processes.

A *software production process*, building from the Feiler and Humphrey definition, can be characterised as a partially ordered network of interacting *activities* aimed at the production of a *software product*. The software production process needs to be *modelled* (that is explicitly represented) in order to be effectively repeatable, re-usable with some specialisation, automatable and manageable. These are characteristics of all processes, of course, and hence the generic applicability of SPM. In particular we wish to be able to both *control* and *support* the software production process.

Both the actual process and the corresponding *process model* need to be able to be evolved in order to cope with change both to the product being produced (and maintained) and to the process (and supporting tools) used to produce it.

Thus a software production process is part of a wider process which we term the *software process* which includes the *meta-process* managing the evolution of the production process.

This notion of a *meta-process* is of fundamental concern to the SPM community and is perhaps one of the distinguishing features of SPM when compared with BPR. This is possibly because BPR does not seem to be generally concerned with evolution; this issue being addressed by initiatives such as TQM.

4 On synergies and pitfalls

4.1 Process Models

The clearest, and cleanest link, is that both disciplines require that a Process be formally described either as part of *re-engineering* or before it can be *enacted*. The form of description is a matter of taste and application domain. However

there is a need to repeat previous warnings that we are dealing with the most difficult type of model and also, because of their very nature, these process models of systems are strategic and even life-threatening. (Or even in this modern market economy, business threatening..!).

Their are two problems which have always been with us but are exaggerated in this context.

- Process models need to be read by non-computer scientists. Their purpose, certainly in a BPR context, is to enlighten management; so they need to be accessible to management and to express the process in management terms.
- They are used to create *process programs* so they need to be expressed in a way that will aid the program developer. In particular it is inevitable that the Process Modelling Language style will intrude. If, for example, the language is object-oriented then the model will reflect this style.

These two objectives are probably mutually exclusive so we need multiple but consistent views of a process.

4.2 Process-Centered Environments

The second and obvious link is that both require an environment to support Process Enactment. There are two distinct architectures emerging:

- Workflow Technology: in essence built around the notion of the management of office entities. Letters can be authorised, posted etc under the control of a letter-management process.
- Process Technology: built around the concept of people performing *roles*: *process* management with the entities being of secondary significance.

Time will determine whether both are necessary or whether one is more sensible than the other. The central issue is that of the handling of change, that of process evolution.

Essentially processes must evolve. The introduction of any form of process mechanisation changes the process. Thus, by definition, each mechanisation must be performed at least twice and then the changed process changes the process....

In Process Modelling systems this problem is addressed by the provision of a persistent store containing some representation of the enacting process plus the ability to dynamically re-bind to this persistent store a new or modified process model fragment. There are essentially two approaches:

- To provide a persistent database and provide programming support exploiting dynamic joins etc.
- To provide a persistent language in which the compiler run-time store management becomes the persistent database.

My own preference is for the latter and that is what is implemented by the IPSE 2.5 system [4], now marketed by ICL as ProcessWise Integrator. The

reason being that this gives full run-time type checking plus a generic ability to re-bind process fragments through the compiler system. On the other hand most existing systems have chosen the former approach probably because of the ready availability of database systems and the relative paucity of persistent language implementations.

In most Workflow systems the approach is to stop the system and re-install an enhanced version. This must be of concern as the systems get larger and older. How old is your oldest 'email' message?

An added difficulty is that persistent systems persist. So we could end up with not only computerised bureaucracy but the persistent record of its functioning.

4.3 On pitfalls

One should always be suspicious when faced with coincidencies. To a scientist they offer opportunity but also the possibility that two plus two makes five - always an enticing and irresistible prospect however fallacious.

The answer, as always, is to properly seperate concerns.

- Its very sensible to analyse the business processes which determine how a business functions. What else would one ever do?
- Its very sensible to integrate the IT strategy with the business strategy. After all knowledge is power and IT exploitation is clearly a major factor in competitiveness. Especially with the growth of electronic trading, video on demand services etc, there is no doubt that all companies will need to offer different services to remain competitive.
- Its very sensible to introduce the notion of process enactment as the basis for the support of collaborative work.
- It is not sensible to design the IT system using the same techniques as for designing the business. Its the same trap as that of imagining that a process model *is* the process. In practise the process model turns out to be a very useful technique for defining the interface *between* the IT system and the business system, but that is quite different from a unified approach.

The co-ordination system is essentially a software system and needs to be designed as such. The fact that it uses familiar social notions such as Process, Roles and Interactions should not lead us to suppose that the words have the same meaning in both contexts.

In fact from the viewpoint of seperating the domains of (after Dowson [5])

- process production (creation of the model),
- process enactment (the execution of a process model)
- process performance (the carrying out of the real-world process)

the usage of the same terms is rather misleading.

Dowson goes to great lengths to show that these domains must be treated, to an extent, independently and that differing tools and techniques are needed

to support each domain. Hence the previously mentioned emphasis on *meta-process* which is the means of a seperation of concerns between the process used to manage the evolution of the process model itself and the real-world process being supported by the process model.

In particular that we should not confuse the enaction of a process model with the actual carrying out of some real world process. This approach argues that we should be very careful when we discuss, in a BPR context, the *integration* of the IT system with the business system. Rather we should discuss the *interface* between the IT system and the business system recognising that each has its own rules and structures.

The difficulty arises because we are using Process Models to perform three functions:

- Firstly, in order to integrate the IT strategy with the business strategy we need a model of the business processes to be supported. Process Models in this case are a representation of the business.
- Secondly, in order to integrate a set of software tools to support these business processes, we need a *Process Architecture*. That is a conceptual framework for consistently integrating a set of process fragments. Process Models in this case represent the interface between the business system and the IT system.
- Lastly, there is a need to provide support for co-operating workgroups. Process Models in this case act as the means of describing coordination between people.

This overloading of the use of process models is both a good thing and a bad thing. Its merits are that it suggests a unified approach to all of these issues. Its drawbacks are that all of these aspects of systems are still in their relative infancy and integrating three research areas into one generic search for a unified theory is a very risky undertaking.

5 Conclusion

We should appreciate that there are many dimensions to the *Process Modelling* domain and that a seperation of concerns is clearly needed in order to individually mature the various aspects.

In particular we should be very clear when creating a Process Model in either the BPR or SPM domain that the reasons for the model are well defined. The previously described differing uses of Process Models require different approaches and representations and the appropriate method must be selected for each type of application.

So what will be the important enduring result of all this focus on process? My own opinion is that it will be in an enhanced ability to mediate between the design of the business system and the design of the IT system rather than a revolutionary approach to organisational design.

References

1. Feiler, P and Humphrey, W. "Software Process Development and Enactment: Concepts and Definitions." Procs ICSP 1993.
2. J.N.Buxton, J.N and Randell, B. "Software Engineering Techniques" Report on NATO Science Conference October 1969
3. Hammer, M. "Re-engineering Work: Don't Automate, Obliterate!" Harvard Business Review July 1990.
4. Warboys, B.C "The IPSE 2.5 Project: Process Modelling as the basis for a Support Environment", Procs First International Conference on Software Development, Environments and Factories, Berlin, 1989.
5. Dowson, M and Fernstrom, C. "Towards Requirements for Enactment mechanisms", Procs 3rd European workshop on Software Process Technology, Springer-Verlag LNCS 772.