# An evolutionary approach to process engineering

I. Robertson,
Informatics Process Group
Department of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL, U.K.
ir@cs.man.ac.uk
http://www.cs.man.ac.uk/ipg/

**Abstract**

This work proposes an adaptation of the conventional view of engineering to account for the unique problems associated with the representation, analysis, and enactment of models of organization processes. The particular factors required of evolution support processes are identified, and experiments to confirm abstract concepts are described.

Accepted for: Many Facets of Process Engineering Workshop, Tunisia Sept .

# An evolutionary approach to process engineering

I. Robertson,
Informatics Process Group
Department of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL, U.K.
ir@cs.man.ac.uk
http://www.cs.man.ac.uk/ipg/

**Résumé**

*Dans ce travail, nous nous proposons de faire une adaptation du point de vue conventionnel en ingénierie pour rendre compte des problèmes uniques associés à la représentation, à l'analyse et à l'exécution des modèles des procédés d'organisation. Les facteurs particuliers requis pour le support des processus d'évolution sont identifiés et des expérimentations confirmant les concepts théoriques sont décrites.*

**Abstract**

*This work proposes an adaptation of the conventional view of engineering to account for the unique problems associated with the representation, analysis, and enactment of models of organization processes. The particular factors required of evolution support processes are identified, and experiments to confirm abstract concepts are described.*

Keywords: process, modelling, modeling, engineering, evolution.

## 1. Introduction

The 'engineering' approach has been extremely successful at addressing practical problems arising in society. Its basis in Technical Rationality, the belief that empirical science is the only source of positive knowledge and that technology is both political and moral, has been a successful frame of reference for applying the results of science to the improvement of the human condition.

This was the original approach adopted (naturally) as the way to deal with the problems posed in the domain of process modelling. Unfortunately the results, particularly for enactable support systems such as process-centred software engineering environments, have been un-spectacular and this work questions the usefulness of this traditional approach and proposes an alternative which is based on the observed features of evolutionary systems.

The following section describes the characteristics of the engineering method, and then some of its limitations are highlit in Section 3. Section 4 describes an adaptation of this traditional approach which may be more suited to the domain of process modelling. It is concerned with the managing of problems (i.e. minimising them) rather than the solving of problems (i.e. eradicating them). Some conclusions are drawn in Section 5.

## 2. The engineering approach

In trying to understand what we mean by the term 'engineering', definitions can be unhelpful. For example, the IEEE defines [IEEE, 1990] software engineering as 'The application of systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is the application of engineering to software'.

Simon [Simon, 1969] provides a better starting point: 'Science is concerned with understanding the natural world, and engineering with intervening in the natural world with artificial constructs for human benefit'.

These interventions are artefacts which change the state of the real world. Engineers bring about that transformation and, once it is effected, their task is very largely finished. Why is it necessary to intervene in the natural world? Because there exists a problem which we believe we have the necessary skills, knowledge and resources to address. It can be simply to 'make the world a better place'.

The 'problem', or the issue to be addressed, might be the cause of some difficulty, but it can also mean the challenge of an opportunity. The issues can best be understood as inconsistencies. They are inconsistencies between the real

world as perceived, and some mental model of how that reality could be. The inconsistency can be a problem (e.g. too many people dying because of polluted water) or an opportunity (e.g. people would pay a premium for crossing the Atlantic more quickly). Of course many inconsistencies of which we are aware have simply to be endured, because we are not in a position to address them. The concept of inconsistency will be referred to again later.

The term 'engineering' thus relates to a kind of problem-solving by construction of artificial artifacts (both physical and abstract). It is used to adapt nature to suit people, but in so doing it may change people's behaviour. It produces things as opposed to experiences. It is about making best use of the knowledge that we have, albeit imperfect, incomplete and uncertain, and finding safe, pragmatic and economic solutions within known constraints.

The engineering of processes is a way of solving problems arising in the context of organizations, in particular those organizational problems whose solution lies in exploiting the use of process models and possibly also co-ordinative and integrative technologies. As with other problems suitable for engineering-type solutions, a problem statement can make specific reference to a process, or it may only identify symptoms from which the true problem(s) may have to be deduced, and which must in turn be addressed. The common concern has a clear focus in the identification and definition of a problem. The intervention in the real world of the organization is a process model or an enactment system, and traditional engineering assumes that once they are implemented the problem or inconsistency will disappear. Unfortunately, this does not often happen.

## 3. Limitations

The engineering intervention is conventionally accomplished in three distinct steps: understanding the problem, designing a solution (the Oxford English Dictionary definition of which is to 'contrive', or 'invent'), and the implementation of that solution in the real world. This model assumes the pre-condition that the requirements (statements of what is needed to solve the problem) are complete, correct and consistent. This is rarely the case, but experience has shown that the consequences of this can be handled in the domains of traditional engineering. Engineering subsequent to implementation is generally confined to a relatively minor activity called 'maintenance'. Artefacts produced by traditional engineering were generally incapable of adaptation, and evolved not through maintenance activity but through new generations of artefacts. It was the design that evolved, not the artefact implemented from the design.

With the development of 'soft' engineering (such as software systems, and later process models), this maintenance activity became much more significant and important, and is characterized by its inadequacy. There are many reasons for this [Lehman, 1987, Perry, 1994]:

- The phenomenal rate of development of new tools and support technologies promotes premature obsolescence and the burden of legacy systems.
- Rapidly changing views on organization purpose and structures has shortened the 'life expectancy' of established business processes.
- The pool of skills available to organizations can change in unforeseen ways.
- These factors greatly increase the difficulty of attempting to specify complete, correct and consistent requirements for process models, or indeed, of software systems.
- The realization that the development process is, in some respects, one of mutual learning between developer and user [Hirscheim and Klein, 1989] means that full requirements may never be established.
- The products of soft engineering are much more malleable than that of hard engineering, and users wish to exploit this.
- Businesses have less time to respond to changes in their market.

There is a need for engineers to make the subject of their designs, both process models and their support systems, capable of evolving timeously in response to that rapidly changing environment.

## 4. The evolutionary approach

The artefacts produced as a result of engineering activity frequently remain in existence for a very long time. For some kinds of artefact, this gives rise to unique problems. Artefacts such as software systems, and now process model instances, have a potentially long life and, problems arise about how to continuously adapt them as their environments change over time. The relief from inconsistency (between real and modelled worlds) provided by the original intervention is only temporary: in a very short period, new inconsistencies appear and a new variant of the old problem emerges. With soft engineering products, inconsistencies can remain even after implementation and in fact may even become more significant. They might become more significant as users obtain a better understanding of their process and so consequently wish to change the process model, as misconceptions of the original process developers emerge or because of a desire exploit new technological developments.

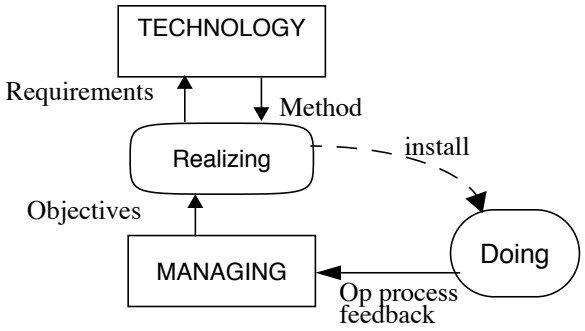


Figure 1: The Process for Process Evolution ($P^2E$)

The concept of evolution is well established as one of the paradigms by which the design process can be understood. That kind of evolution, 'ontogenic design evolution'[1], is the process through which the design unfolds from it initial form, to the form in which it is to be implemented. In traditional engineering, and some soft engineering, product evolution is manifested in new generations of artefacts. The motor car is replaced by a newer and better model: the IT system is upgraded by replacing its instance with a new one. What has been of less significance (from an engineering viewpoint) is evolution of the artefact itself, so-called 'phylogenic'[2] evolution. Phylogenic evolution occurs when there is a misfit between the implemented design, say of the process model, and its operational environment, and is manifested as a change in requirements. The misfit is eliminated by re-design based on these new requirements and by adapting the process model instance to correspond to the new design. This is what is conventionally referred to as 'maintenance'. It has always been viewed as an evolutionary process by practitioners [Arthur, 1988], but whereas it was formerly concerned with relatively minor changes to the design and implementation of an artefact, it must now be seen as a major aspect of the design solution to the original problem. The basic form of the process model may now be the subject of change.

In the context of process modelling, the need to change after implementation has been well understood for many years [Conradi et al., 1992] and the concept of the meta-process is well established. What has possibly not been apparent is that the meta-process needs to be an evolutionary process, in fact a process of artificial (purposeful) evolution.

A study of literature has related the characteristics of naturally evolving systems to those of the domain of process engineering. It suggests that, for a process to be effective in supporting true evolution (as opposed to one which supports only change) it might have to posses certain properties:

- There is an identifiable object (which is the system that evolves) and an environment (the context in which that evolution take place) [Sagasti, 1970]. For example: are the developers part of the object system or part of the environment of that system?
- That object must be capable of adaptation.
- There is provision for some kind of testing of the object in the environment (to determine its 'fitness' in its environment). Testing is not the prerogative of developers, but is also a concern of users themselves.
- There is provision for the feedback of test results ('misfits') and support for their evaluation [Dasgupta, 1991, Lehman, 1987]. This may be by means of formal test reports, or by suggestions or complaints from users.
- There is a way of eliminating misfit, if that is what is desired. For example there may be a process in the organization which collects test reports, assesses them, and translates them into a set of objectives. From these a set of requirements may be established, a method for addressing these developed (in the form of a process model), and then implemented.

The purpose of evolution is to eliminate misfit between the object system and its environment. Unfortunately the environment is not static. It is composed of other objects which are in themselves being adapted to their environment. This is co-evolution. The consequence of this is that the engineering design process has to continue after implementation. In essence, the engineering problem situation becomes one of how to maintain the inconsistency (between real and model worlds) *within reasonable bounds, over time* [Arbouai and Oquendo, 1993]. This may be compared with the traditional engineering approach of eliminating this inconsistency by implementing the product of a development project. If the inconsistency is eliminated, the problem is solved. The place for this, then, is in a meta-process which ought to reflect these properties.

---

1. See [Dasgupta, 1991]
2. Op cit

A process model possessing these properties has been developed by the IPG and is fully described in [Warboys, 1997]. This is the Process for Process Evolution (referred to as $P^2E$) (see Figure 1). It is a behavioural model based on a separation of concerns. One concern is 'Managing' which encompasses activities associated with organization direction, purpose, and relating visions to reality. 'Technology' is concerned with finding methods (in the form of generic process solutions), by means of which the management vision (encapsulated in some statement of Objectives) can become a reality either by re-using an earlier solution, or by developing a new one. 'Realizing' is concerned with changing reality. It does this by first of all tailoring the generic process solution (the method) to the specific details of both the objectives that have to be satisfied, and the organization in which this is to be done. Once the process has been tailored, it has to be installed in the organization and put into effect. The concern of 'Doing' is the undertaking or the enacting of the processes in the real world, for example it could be a software production process. In Figure 1, solid arrows indicate information flow, and the broken arrow indicates the application of behavioural change.

Realizing and Technology together account for the conventional concerns of engineering: the establishment of requirements, the seeking of a solution (the design activity), and the implementation of that solution in the real world.

This model can, when implemented in a suitable technology, support these features which must exist if evolution is to be facilitated. It has in fact already been implemented in the ProcessWeb [Greenwood and Warboys, 1996, Yeomans, 1996] process support system and can demonstrably [Robertson, 1996] support the evolution of a single process.

The challenge is to adapt this model (or similar ones) to support the complexities and conflicts of the real world.

## 5. Conclusion

The classic engineering approach to the solution of problems in process modelling might no longer be appropriate. The alternative approach proposed in this paper is, rather than solving specific problems, to appreciate that a single intervention will not be sufficient. Inconsistencies between real and model worlds will quickly re-appear because of co-evolution, so there is a need to manage this inconsistency in order to keep it to a minimum, which in turn means that support should be provided for continuous interventions. Consistency between real and model worlds is a goal to be aimed for but it is unlikely ever to be realized. Even if it is achieved, it will only be short-lived because the ever-changing environment will cause new inconsistencies to become manifest.

A model (the Process for Process Evolution) has been described which represents the behaviour associated with such concerns and this model is demonstrably capable of being implemented. It provides, at the very least, a management framework to deal with inconsistencies, with a view to minimising their magnitude and duration, by allowing the implemented process model to evolve in a systematic manner.

Once a process model is in existence, new requirements for the process result in a new version of the design for the process model. This new version is not implemented by means of a new instance of the model, but in adapting the existing instance to comply with the new version of the design. An example of this kind of on-the-fly adaptation has been described elsewhere [Robertson, 1996].

It has been said [Conradi et al., 1992] that every process model needs a meta-process model, however the scope of possible change is an open issue and in fact it is likely that each process fragment, or component, ought to be bound to its own meta-process. The problem then becomes one of managing the meta-processes. The IPG will be developing a new modelling and enactment language as part of the Compliant Systems Architecture project at Manchester, which has recently commenced. One of the language features under consideration is to provide a $P^2E$ component as one of the primitives of the language, thus enabling us to enact models comprised of evolvable components.

Evolution has been discussed in the context of a single process model object existing in an environment, but of course the environment is partly made up of other process objects that are also evolving. Such co-evolution is a reality and process models, if they are to be successful, will have to reflect this. The IPG believes that it is possible to demonstrate co-evolution by making use of one of the products of the Evolution of Large Software Systems (ELSS) project [Greenwood et al., 1996, Patel, 1996], the product tower, to show how two or more software systems can evolve independently but in relation to the other system. It is also hoped that the IPG's new ProcessWeb system can be cast as a product tower to facilitate it's future evolutionary development.

The original Process Engineering Framework (PEF) Methodology, developed by the IPG [Wastell et al., 1994], is based on establishing relationships between business goals and process interactions. It is now recognized as being inadequate for dealing with dynamic conditions that commonly exist in the real world, and that if it is to be effective, then it must support evolution. Work revising its focus will begin later this year when the new PELSIAM project (Process Evolution and Legacy Systems Integration using Active Models) gets under way at Manchester.

## Acknowledgements

The author is indebted to his colleagues in the Informatics Process Group and to those in the Esprit PROMOTER working group who contributed both valuable ideas and criticism to this important area of inquiry.

This work was made possible by the EPSRC funded APPUS project (Assisting Process Performance and Utilisation through Simulation).

## References

Arbouai, S. and Oquendo, F. (1993). Managing Inconsistencies between Process Enactment and Process Performance States. In *Proceedings 8th International Software Process Workshop*. IEE Computer Society Press.

Arthur, L. (1988). *Software Evolution*. John Wiley and Sons.

Conradi, R., Fernström, C., Fuggetta, A., and Snowdon, R. (1992). Towards a Reference Framework for Process Concepts. In Derniame, J.-C., editor, *Proceedings 2nd. European Workshop on Software Process Technology EWSPT'92*, volume 635 of *Lecture Notes in Computer Science*. Springer Verlag.

Dasgupta, A. (1991). *Design Theory and Computer Science*. Cambridge Tracts in Theoretical Compter Science. Cambridge University Press, Cambridge.

Greenwood, R., Sa, J., and Warboys, B. (1996). Cooperating Evolving Components - a formal approach to evolving large software systems. In *Proceedings ICSE'96*.

Greenwood, R. and Warboys, B. (1996). ProcessWeb - Process Support for the World Wide Web. In Montangero, C., editor, *Proceedings of the 5th. European Workshop on Process Technology (EWSPT'96)*, volume 1149 of *Lecture Notes in Computer Science*. Springer Verlag.

Hirscheim, R. and Klein, H. (1989). Four Paradigms of Information Systems Development. *Communications of the ACM*, 32(10).

IEEE (1990). Standard Glossary of Software Engineering Terminology. ANSI/IEEE Std. 610.12-1990, IEEE Piscataway, NJ.

Lehman, M. (1987). Evolution of Process Models, Process Programs, Programming Support. In *Proceedings 9th. International Conference on Software Engineering*. IEEE CS Press.

Patel, D. (1996). A Process for Evolving Large Software Systems. Master's thesis, University of Manchester.

Perry, D. (1994). Dimensions of Software Evolution. In Müller, H. and Georges, M., editors, *Proceedings International Conference on Software Maintenance*. IEE Computer Society Press.

Robertson, I. (1996). An Implementable Meta-process. In Tanik, M., Bastani, F., Gibson, D., and Fielding, P., editors, *Proceedings 2nd World Conference on Integrated Design and Process Technology*, Austin, Texas. Society for Design and Process Science, 1301 West 25th Street, Suite 300, Austin, Texas 78705-4236, USA.

Sagasti, F. (1970). A conceptual and taxonomic framework for the analysis of adaptive behaviour. *General Systems*, 25.

Simon, H. (1969). *The Sciences of the Artificial*. MIT Press.

Warboys, B., editor (1997). *Business Systems Engineering: A Process Approach*. In preparation.

Wastell, D., White, P., and Kawalek, P. (1994). A methodology for business process re-design: experiences and issues. *Journal of Strategic Information Systems*, 3(1):23–40.

Yeomans, B. (1996). Enhancing the World-wide-web. Project report, Department of Computer Science, University of Manchester http://r305.cs.man.ac.uk/.