24

An introduction to a process engineering approach and a case study illustration of its utility

P. Kawalek,
Informatics Process Group, Department of Computer Science,
University of Manchester, Oxford Road, Manchester M13 9PL. England.
Tel: +44 161 275 6183. Fax: +44 161 275 6200. email:pk@cs.man.ac.uk

Abstract.
This paper introduces an emergent process oriented approach to systems engineering. It has come to be known within its domain of development as 'process engineering.' The approach has been developed through collaboration with companies in the services and manufacturing sectors, and through the 'Process Engineering Framework' Project. A case study is used to illustrate the approach which relates to the development of software.

Keywords.
Process engineering, process modelling, process architecture.

1 INTRODUCTION.

1.1 Aims

The subject of this paper is a process engineering approach which is known as the Process Engineering Framework (PEF) after the UK EPSRC project which is developing it. As an introduction to the approach this paper aims to address its nature, position it in relation to related approaches and present two of the most important topic areas within it. These are modelling and architecture.

1.2 The orientation of the approach

At some risk of over generalisation it is suggested that systems engineering is still maturing as a systems discipline. One is reminded of how systems theory was born as biologists in particular faced problems of complexity and order. The sort of problem encountered was

J. Browne et al. (eds.), *Re-engineering the Enterprise*

summarised by Weiss (1968, p.22) who described how in analysing the cell the enquirer encountered "....rather well-defined and relatively stable complexes of functional and structural properties which are embedded in, and mutually related through, matrices of much less well-defined, more fleeting configurations..." The concept of 'system' is used to understand and describe the properties that parts have when in relationship with other parts. The concept has been defined with differing emphases many times, for example by von Bertalanffy (1968, p.55), Checkland (1981, p.317), and Beishon and Peters (1976 p. 12). Checkland is noted for applying 'soft' systems theory to the study of social problem situations. Leaving to one side the main thrust of his argument, it is useful to note his characterisation of systems through two pairs of ideas; control and communication, and hierarchy and emergence. For the details of this refer to Checkland (1981, pp. 74-92). Although these ideas are not of direct concern in this paper, the reader may note how they recur in the later discussions. More directly relevant to our purpose at this present time is another systems concept, that of systems synthesis. Systems synthesis can be described as concerned with the understanding of the larger system to which a part belongs, appreciation of the properties of this larger system and then recognition of the role played by the part within this larger system (Wardman, 1994). Or, as Weiss puts it "By raising his sights from single objects to their interrelations with others, man reverses his direction from analysis to synthesis" (Weiss, 1968, p.6).

An example

A recent project undertaken in collaboration with an insurance company can be used to describe a practical manifestation of analysis/synthesis. The company have invested heavily in two world-wide databases, one for in progress work and one for active policies. However, currently their business processes are characteristically manual. Most work is done using bulky paper files which contain the policy submissions received from brokers. The files are used by underwriters who scour the details of the submission as part of their risk assessment. They add hand-written notes to the files which describe the reasoning which underpins their decisions about premium quotes and so forth. Administrators use paper forms upon which they hand write summary details of each submission. These are attached to the files. At specified points in the process the information contained in these forms is transferred to a database. Cases which progress to the successful negotiation of a premium will be entered into both databases thereby requiring duplication of effort. Locally, in order to remedy specific information shortages, initiatives have resulted in the development of new databases (e.g. containing brokers names) using systems such as Paradox. Although these do help to remedy the information shortages, they add to the problem of repeated data entries. They require that additional actions are carried out by users so as to keep the databases up to date and in accord with other databases. Therefore, certain items of information are entered in triplicate to databases and at least once to paper forms. This is an example of succeeding with analysis but failing with synthesis. That is, although each of the databases might of themselves be well designed and able to satisfy some requirement, when we consider how the IT system as a whole serves the company's staff it is apparent that the relationship is not satisfactory. The desire to be served by a system which requires data items to be entered once and once only was a something of a leitmotif for the study.

The emphasis in systems engineering today

Returning to systems engineering, and considering it from an organisational perspective, there is evidence that the discipline can be characterised by an increasing concern for problems of synthesis rather than classical problems of analysis (e.g. the bespoke development of a payroll system). The increased emphasis given to systems integration, business led design approaches, holistic process reengineering approaches, workflow, business network redesign (Venkatraman,1991) and to some degree networking more generally, all seem to testify to this increased concern for synthesis. This is reinforced by the aspiration for reuse of components in a lego brick style of development. The approach described in this paper uses ideas from process modelling to explore, define and evolve the relationship between parts of a system. At one level these parts can be seen as the social and technical systems within an organisation, and at another they might be the individual people working for the organisation and their various tools (e.g. databases).

1.3 Process Engineering and Business Process Reengineering.

In anticipation of some terminological difficulties this part of the paper seeks to separate PEF from the vaunted Business Process Reengineering (BPR) which, whilst sharing some of the concerns of PEF, has different origins and emphases. The reader is asked to set aside any preconceptions he or she has about 'process engineering,' 'Business Process Reengineering,' 'process modelling' and the like, and instead think simply of a sociotechnical system. Very simply, the social system can be characterised as being made up of people and their concerns (including culture, politics and structures). An equally simple characterisation of the technical system might describe it as being made up of tools such as the computer systems that are ubiquitous in modern enterprises. These systems can be described as having a relationship in that people in the social system use tools in the technical system to carry out actions. A starting position is then that the relationship between these domains can be understood by analysis and synthesis of actions. In simple terms a process is made up of a number of actions which serve an objective and so we can suggest that the relationship between a social system and a technical system can be understood by exploration of a process which people seek to carry out. Both BPR and PEF use the concept of process dually and simultaneously to see how actions are related to each other and to understand the relationship between social and technical systems.

This said, and although one of the often expressed concerns of BPR is to achieve maximal benefit from IT (Hammer, 1990), characteristically its subject matter is the social domain. Its maxim seems to be that once the capabilities of the technical system are understood, radical but bountiful changes should be made to the social system to exploit these capabilities. In the various writings there seems to be little attention given to the form of the technical system itself (see for example Hammer and Champy, 1993). PEF is different because by origin and nature its primary subject matter is the technical system. More specifically still its domain is informatics (which is made up of information and telecommunication technologies). It draws upon research into the development of support environments which at a minimum level provide integrative mechanisms between defined sets of tools for users. The emphasis today is upon the use of modelling and architecture to develop the relationship between social and technical systems. These topics of modelling and architecture will be considered in more detail later.

The conceptual separation of the social and the technical is very useful and is used for a number of reasons in a number of circumstances. Here it has been used to characterise BPR as concerned with the form of the social system and PEF as concerned with the form of the technical. However, we must appreciate that in a deeper sense the social and technical are not separate at all. The form of the social system is bound up with the form of the technical system and vice versa. In our work we have appreciated this and have extended our methodological concerns to analysis and development of the form of the social system (Wastell et al., 1994). This work, though not the primary focus of PEF, shall continue as we appreciate that the development of the technical system is intimately bound up with the development of the organisation in which it sits.

Finally, although it has been alluded to already it is important to recognise explicitly the characteristic holism which is shared by BPR and PEF. This like many of the points made in this paper relates to the idea of systems synthesis which was introduced earlier. In BPR the emphasis is upon processes which cross functional boundaries, which reach from customer request to customer satisfaction and facilitate change programmes from an enterprise perspective rather than from a organisation function perspective (Hammer and Champy, 1993, p. 35). In PEF the messages are characteristically more technical; that process is a way of integrating roles, and that the various transactions between tools and roles that are undergone in satisfying a customer need can be understood as a 'long transaction.'

2 MODELLING.

2.1 Introduction

Although the focus is upon the form of the technical system, the process engineering approach of PEF starts with models of process in the social system that it seeks to serve. Hierarchical and coordinative models are both used. Hierarchical models are useful because they allow us to develop control structures which are important for process change. They are not dealt with in this paper. Coordinative models are useful because they allow us to explore the composition of components in a process and the communication between these components. These models may represent an existing situation or a desired one. The difference is obviously critical in real world projects and for the purposes of this paper it is helpful if you bear in mind that all the examples presented are a design for an as yet unrealised process.

The case study

The examples in this paper result from an evolutionary, non-radical redesign exercise undertaken by the author. The project was undertaken in collaboration with a British company who operate a number of large chemical plants. The subject of the study was a software development team who are located at one of these plants. There are sixteen software engineers divided into two groups. Most of the work is concerned with the development of software for complex instruments used around the plant. Some of these instruments are classified 'safety related' by the EC and many more are in other ways critical to the operation of the plant. It is therefore imperative that the team are able to develop high quality software and the team have a very good track record in this respect. The project is motivated by three

goals in particular. First, it seeks to evolve the social system/technical system relationship in order that certain minor frustrations such as the need to discontinuously maintain a quality tracking system are overcome. Secondly, it seeks to allow a greater degree of flexibility over the way in which certain, non-safety related projects are currently handled. Thirdly, recognising that the organisational and economic circumstances of the team are changing, the project is concerned with making the process easier to evolve in the face of future, as yet unknown circumstances.

2.1 Coordinative Models.

In order to understand the relationships between components of a system, we need to be able to express what these components are and how they interact with each other. The following model is a fragment of the design for a new process. It shows the components of a process and the interactions between these components.

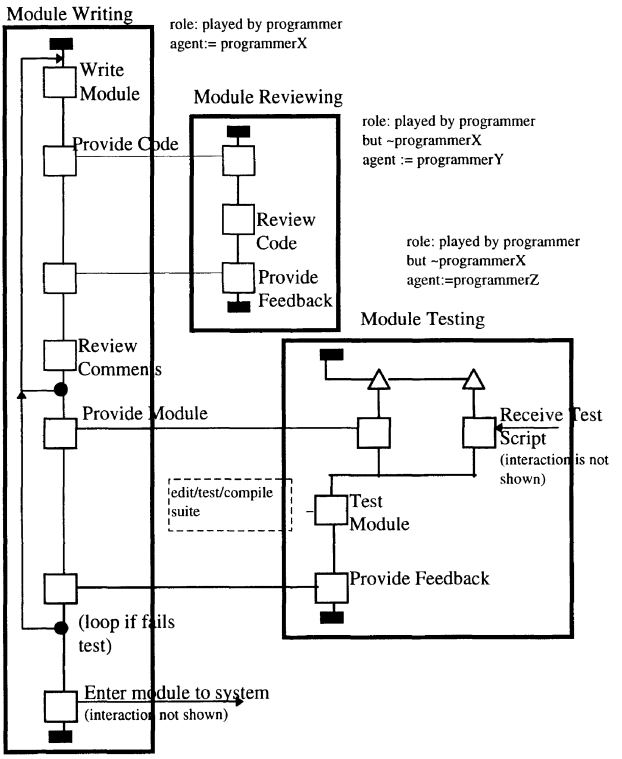

Figure 1 An example of a Role Activity Diagram.

The notation used in this example is the Role Activity Diagram (RAD). This originates from the work of Holt et al. (1983). There is not yet a RAD standard although a complete definition has been given by Ould, (1992). A RAD is a state based diagram in which the vertical lines between boxes represents different states. The boxes are actions. Interactions between roles are a special kind of action and are identified by a horizontal line linking each end. Where one end of an interaction is not shown the horizontal line has an arrow head indicating direction. The triangles represent the commencement of a thread in a part refinement. Each thread in a part refinement can be thought of as a sub-state to the main thread. Therefore as this implies there is no ordering between threads within a part refinement. In the above example loops are allowed. These are lines with arrow heads and a black circle at their commencement. The start and finish of each role is marked by a black rectangle. Finally, the role itself is represented by the rectangle grouping around an action thread (i.e. 'Module Writing,' 'Module Reviewing' and 'Module Testing'). Roles are made up of actions which are related by the internal structure of the role, and thereby the role has the property of emergence. Ould has defined a role as "..a set of activities which, taken together, achieve some particular goal" (Ould, 1992). Roles are played by people and so we might speculate that roles for a lecturer might include 'lecture giving,' 'notes preparing,' 'research direction setting' (gerunds are the convention). Although there are good reasons for wanting a more rigorous definition than this within PEF, this will do for our purposes in this paper. All the roles in this fragment are undertaken by programmers. There are rules which require that the same programmer does not write and review the same module or that he/she does not write and test the same module. These rules are shown as annotations on the diagram. There is one other annotation which shows that the tools which make up the 'edit/test/compile suite' are used in the 'Test Module' action. Annotations which show the technology required by actions are important. Each action shown in the diagram could have been annotated this way. Only one example is given in the example because of the problem of clutter in the restricted confines of the page.

In the RAD, as well as actions, structuring of actions and interactions we see information about role players and technology. Fully annotated it would be a map of a social system/ technical system relationship. We see person to person and person to technology relationships. Technology to technology relationships can also occur although they do not in this fragment. RADs are valuable as representations of current process or as blueprints of process designs. They show co-ordination within and between roles. It is argued that their production and reading can help develop shared understanding amongst individuals involved in a study, that they can help make covert problems manifest and can help rationalise process and procedure (Ould, 1992), (Kawalek, 1994). All of this is important. However, perhaps more interesting is to ask what we could have if the RAD were not just a passive model but somehow enactive. What if instead of just being a representation or blueprint, the RAD represented an encoded prescription of the behaviour we want of the technical system? Thus, for the programmer playing the role of 'Module Testing' the system would manage the interactions with other roles to obtain the module and the test script. It would present the programmer with an icon 'Test Module.' On clicking this icon, the tools necessary for the programmer to carry out the action ('test/edit/compile suite') would be presented. The programmer could finish the test and use the system to send the feedback to the role which created the module.

Process modelling languages

This simple sketch of the concept of an enactive model describes the essence of a process modelling language. The development of PEF is particularly closely associated with an early example of this genus which is fittingly known as the Process Modelling Language (PML) (Bruynooghe et al., 1994). PML is a high level language within which the basic building block is a role. PML roles carry out actions and have interactions with other roles. To this extent there is a clear mapping between PML and RAD primitives. PML was developed as the language of the IPSE 2.5 software engineering environment (Warboys, 1991). It has been further developed as the language of the derivative ICL ProcessWise Integrator (PWI) which is used for support of business processes in general. PWI provides a user interface and a PML application interface so that external applications (e.g. 'test/edit/compile suite') can be integrated into the user's on-line work context. The form of this work context (i.e. which actions and technology are available) will vary according to which roles the user plays and the state of the process of which these roles are a part.

2.2 The relationship between passive and enactive models.

The enactive model encoded in a language like PML serves as a centre of co-ordination and control within a system of many different parts. Thus, Snowdon (1992) has talked of using process modelling to develop IT systems from "..an overall systems level." The attraction of using a high level process modelling language lies with the establishment of constructs which serve both analytical purposes in organisational design activity and as a prescription for the behaviour of the technical system. This suggests that the RAD fragment shown previously could be dually a design for organisational behaviour and an enactable prescription for the way in which the technical system will serve the organisation. Thus the requirement for the designer to invert an organisational view in order to consider the design in the language of the IT system, which is essentially a language of calculation, is much reduced or even eradicated.

Methodological issues

It is important to consider the methodological issues involved in the translation of a model from a passive representation to an enactive, working component in the behaviour of an organisation. Is it really possible to take a simple step between passive and enactive domains? The author has observed development practice for PWI and some similar systems. Currently, for a number of reasons including non-functional considerations, the 'translation' between passive and enactive domains is not normally straightforward. Indeed there is a notable distortion of the role structure of the passive model in developing the enactive model. The temptation is to see the development of such systems as more akin to a conventional process through which requirements are met by an (inverted) programming of the system. One significant issue is that passive models created as part of (social system) redesign exercises are not normally intended to be rigid prescriptions of behaviour. They are simplified interpretations of real world behaviour which are usually used to represent a typical case to users. However the reality of work tends to be characterised by exceptions and the need for creative, extemporising behaviour by users (Fikes, 1982). Even in the very simple RAD fragment which was shown previously we can see how, interpreting the model strictly (as software would do), the 'Module Writing' role will have to undertake and complete the action 'Write Module'

before passing the code for review by 'Module Reviewing.' This may be perfectly reasonable for many cases but it precludes any other behaviour. Perhaps tackling a particularly difficult module the 'Module Writing' programmer wishes to half complete it and pass it over to 'Module Reviewing' for initial comments whilst continuing to complete the rest of the module. This is very reasonable behaviour in the actual domain which places a lot of emphasis upon peer reviews and informal support. However, if we were to simply take this RAD and convert it into software the system would not support this reasonable way of working.

This means that we need to explicitly design the behaviour of the enactive model. However, it is the author's contention that the structural integrity of the model should be consistent across passive and enactive domains. This is achieved in the case study by preserving the roles ('Module Writing,' 'Module Reviewing,' and 'Module Testing') and the pre and post-conditions to these roles across the passive and active domains. All that has been changed is the internal structure of the role so that a pleasant and flexible working environment is created for each user. It no longer strictly enforces typical case behaviour but provides a context for flexible handling of many cases. Incidentally, these design decisions were taken in consultation with users and managers. The following example expresses as a RAD the behaviour of the enactive PML model for the role 'Module Writing.' The other roles have been omitted from this diagram.

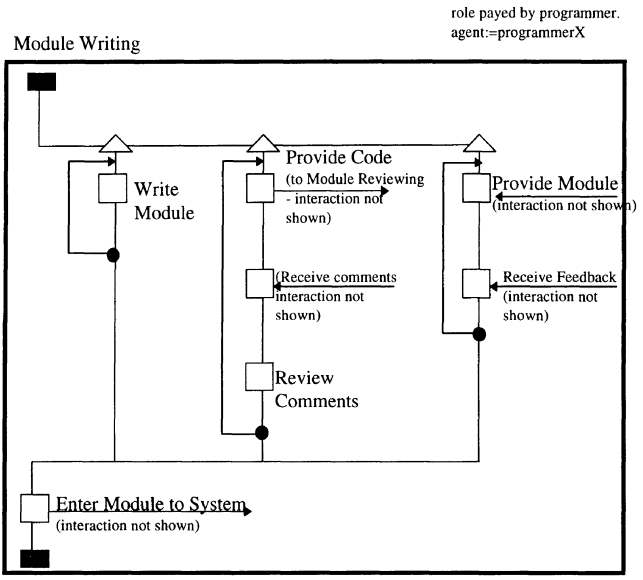

Figure 2 A representation of the behaviour of the enactive model

Thus far we have seen how process modelling can be passive and enactive. It can be described as passive where it is an analytical activity whose subject matter is the form of the social

domain. Passive models are used for development of understanding, representation of complexity and the development of designs. Process models can be described as enactive where they are used in real time to define the behaviour of the technical system and hence the real world relationship between social and technical systems. The simple example of a RAD and PML is used to illustrate how the structural integrity of a model can be maintained across the passive and enactive domains.

3 ARCHITECTURE.

3.1 The enactive model as an architectural component.

It is important to consider the implications of the enactive model for the overall system architecture. How does it affect the form of the system to have this model, which has been described as a kind of hub of co-ordination and control, as a component within it? Although research in this area is still in its very early stages we can speculate that the architectural implications of the active process model could be of profound consequence.

We can start by considering the infrastructure of the software development team in the case study.

- They have a structure of process, roles and actions with associated rules.
- They use non-electronic media such as paper for many informal and some formal purposes.
- They have a number of IT applications, databases, networks and platforms. For purposes of illustration applications of note are the edit/test/compile suite which has its own database, a text editor and a quality tracking database.
- The active process model will be used as a coordinative mechanism and will, when activated influence the way work is performed.

We can explore how the enactive model can serve as a coordinative mechanism between some components in this infrastructure. The effect of this is to enable a greater degree and order of synthesis within the system as a whole. The degree of synthesis that is beneficial is a design issue. The reader should not infer that process engineering will motivate a pendulum like swing from a need for integration to a fully (perhaps overly) integrated system. Indeed, the use of the enactive model to integrate components within a system can be thought of as a way of achieving the loosest possible level of integration between components of the system. From this may arise benefits of flexibility.

The relationship between people and tools.

Repeatedly in this paper two important aspects of the enactive process model have been emphasised. These are discussed by Warboys (1991). The first is that it acts as an "upward facing" framework for supporting the interaction of users in a human organisation. The second is that it acts as a "downward facing " integration framework for disparate tools and databases. The following diagram depicts the enactive model in this role as a coordinative component within a system (social and technical) of people, applications and tools. It was first presented by White and Kawalek (1993).

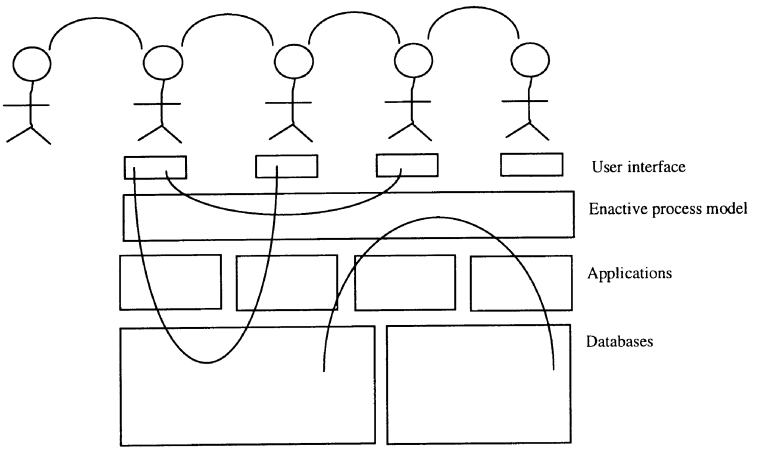

Figure 3 An interpretation of a system architecture incorporating an enactive process model.

The arcs in this diagram represent four different forms of co-ordination. These are as follows;

- Inter-personal (not mediated by IT),
- Inter-personal mediated by the enactive process model,
- Inter-personal mediated by a shared database,
- Between databases (or alternatively between applications) mediated by the enactive model.

It would be valid to denote other forms of co-ordination in this diagram. However those shown are particularly interesting in that we see in points two and four respectively the upward and downwards roles of the enactive process model. The diagram also recognises forms of co-ordination outside of the enactive process model (points one and three).

An example

To illustrate this a small part of the RAD model in Figure 1 is considered. It relates to the interaction between the 'Module Writing' and 'Module Reviewing' roles. Exploring this in more detail shows that the programmer playing the 'Module Writing' role uses the edit/test/compile suite to carry out the action 'Write Module.' The edit/test/compile suite has its own database with version control. It does not share this database with any other application. The 'Provide Code' interaction sends a prompt from the 'Module Writing' role to the 'Module Reviewing' role. On picking up this interaction the 'Module Reviewing' role is connected into the edit/test/compile suite where the module is examined. In carrying out the action 'Review Code' the role has access to a simple text editor for writing a report. All completed reviews must be kept in the Quality Tracking Database. When the review has been completed the programmer playing the role clicks a 'Finished' icon provided by the active

model. The enactive model will then enter the review into the Quality Tracking Database and place a 'Review Comments' icon on the screen of the 'Module Writing' role. When this icon is clicked the programmer concerned will have access to the Quality Tracking Database to read the review. This is expressed in Figure 4.

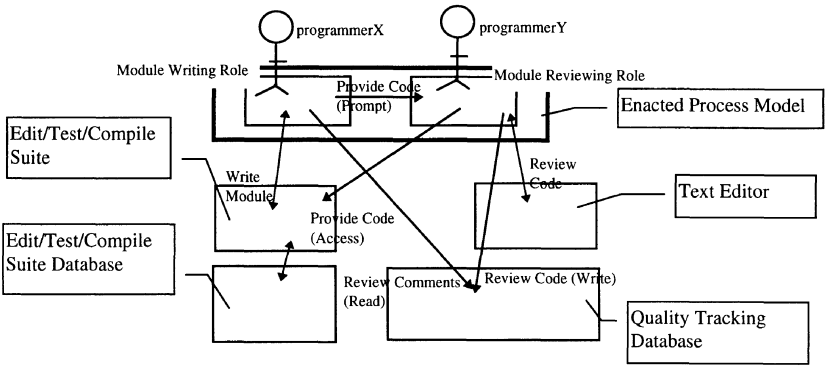

Figure 4 An interpretation of the architectural role of the enacted process model in a small part of the case study example.

Structural holism and functional clarity

This simple example gives us the opportunity to identify two important facets of the PEF approach to architecture. First, it is holistic in a structural sense in that as well as developing models which represent and contribute to the dynamism of human behaviour in organisations, it develops models of the way in which the technical system contributes to the organisation. The approach is concerned simultaneously with interactions supporting the organisation's processes and acting as an integration framework for tools. These are the upwards and downwards roles described by Warboys (1991). Thus we recognise, after Heidegger (1977), that objects such as computer systems become part of a background of 'readiness-to-hand' to their users. The users are not concerned with them as such but with the actions they seek to accomplish. If we wish to consider the actions that people carry out then we have also to consider the tools upon which these actions rely. Warboys (1991) argues that modelling approaches which suggest a separation of the upwards and downwards will have a "...short life."

This holism leaves us with a problem of complexity. One of the ways in which this complexity must be managed is through functional clarity. We have seen earlier how this can be achieved in the RAD models where roles boundaries were defined, their interrelationships mapped and their technology dependencies were annotated. We see it also in the architectural approach whereby we have conceptually separated co-ordination and control from the rest of the system capabilities and classified these other capabilities as applications and databases. Nothing in this architectural approach should be understood to be anything other than an initial response to a very difficult problem.

4 CONCLUSION.

This paper has positioned the PEF approach as concerned with relationships between parts of a system. At its most abstract the relationship of concern is that which exists between social and technical systems. This is worth pondering. This relationship has the potential to become ever more complex. On the one hand a climate of commercial competitiveness and insecurity requires that people and organisations are flexible, creative and innovative. On the other hand the number of technical solutions available in the market-place continues to multiply. Attempts to try and control the import of new tools or the creation of new databases in an organisation are likely to be confounded by the growth of networks and the easy access to hardware. Ultimately the very complexity of the domain may serve to stifle those aspects which are most precious, namely flexibility, creativity and innovation. Any wish to bring structure and simplification to bear will have to start with fundamentals. We need to understand the essential dependencies which bind the work of one person to the work of another. We need to ask what a user wishes to do, what he or she needs from someone else, and what he or she expects to deliver. We can understand these things by understanding process.

5 ACKNOWLEDGEMENTS.

The architecture section "borrows" from a project undertaken jointly with Dick Thomas of Thomas Partnership. Thank you to Mark Greenwood and Martyn Spink for comments.

6 REFERENCES.

Beishon, J. Peters, G., (1976) *Systems Behaviour*, Second Edition, The Open University Press, Harper and Row, London.

Bruynooghe, R.F., Greenwood, R.M., Robertson, I., Sa, J., Warboys, B.C., (1994) PADM: Towards a Total Process Modelling System in *Software Process Modelling and Technology* (ed. Finkelstein, A., Kramer J., Nuseibeh, B.) Research Studies Press Ltd., Taunton.

Checkland, P. (1981) *Systems Thinking Systems Practice*, John Wiley and Sons, Chichester.

Fikes, R.E., (1982) A commitment based framework for describing informal cooperative work, *Cognitive Science,* **6**, 331-347.

Hammer, M., (1990) Reengineering Work: Don't Automate, Obliterate, *Harvard Business Review*, July-August 1990.

Hammer, M., Champy, J., (1993) *Reengineering the corporation, a manifesto for business revolution*, Nicholas Brealey Publishing, London.

Heidegger, M., (1977) *The Question concerning Technology*, Harper and Row, New York,.

Holt, A.W., Ramsey, H.R., Grimes, J.D., (1983) Coordination system technology as the basis for a programming environment, *Electrical Communication* 57(4).

Kawalek, P., (1994) Comments on the use of RADs in case studies, *IOPener*, Newsletter of the IOPT Club, Volume 2, Number 3.

Ould, M, Process Modelling with RADs, Parts 1-3, *IOPener*, Newsletter of the IOPT Club, Volume 1, Number 5 to Volume 2, Number 2.

Snowdon, R.A., (1992) Process modelling is more than sequencing work, *IOPener,* Newsletter of the IOPT Club, Volume 1, Number 3, February 1992.

Venkatraman, N., (1991) IT Induced Business Reconfiguration in The Corporation of the 1990s, *Information Technology and Organizational Transformation* (ed. Scott-Morton, M.S.), Oxford University Press, Oxford.

von Bertalanffy, L. (1968) *General System Theory, Foundations, Development, Applications*, Allen Lane, London.

Warboys, B.C., (1991) The practical application of process modelling, some early reflections, *Proceedings of the First European Workshop on Software Process Technology,* Milan, AICA Press.

Wardman, K.T., (1994) From Mechanistic to Social Systemic Thinking, A digest of a talk by Russell L. Ackoff, *The Systems Thinker*, Volume 5, Number 1, February 1994.

Wastell, D.G., White, P., Kawalek, P., (1994) A Methodology for Business Process Redesign: Experiences and Issues, *Journal of Strategic Information Systems*, Volume 3, Number 1.

Weiss, P.A., (1968) The Living System: Determinism Stratified in *Beyond Determinism: New Perspectives in the Life Sciences* (ed. Koestler, A., Smythies, J.R.) Hutchinson, London.

White, P., Kawalek, P., (1993) A framework for business process management, *IPG report*, Informatics Process Group, Department of Computer Science, University of Manchester.

7 BIOGRAPHY.

Peter Kawalek is a Research Associate of the Informatics Process Group in the Department of Computer Science, University of Manchester. He has undertaken ten collaborative projects with industrial partners. Currently, as well as working on further collaborative projects he is contributing to the EPSRC funded 'Process Engineering Framework.' The results of this project shall be published.