

Behavioural Issues in Software Engineering Process Modelling: A Multi-Paradigm Approach

G. Mike McGrath

*Joint Research centre for Advanced Systems Engineering, Macquarie University, Australia
mmcgrath@mpce.mq.edu.au*

Abstract

There is a growing consensus that if substantial software engineering productivity improvements are to be realised, much greater attention must be paid to the so-called "softer" (or people-centred) factors. A software engineering process modelling framework is presented. The focus of this work is on behavioural aspects.. A feature of the framework is that user views, in a variety of modelling formalisms, may be extracted from a common, highly-abstracted conceptual model. This is consistent with an emerging view that development of more formal models of organization and management theory (OMT) requires a "horses for courses" approach. That is, different theories are often best represented using different paradigms, with the conceptual model serving as a common reference point through which consistent user views may be derived and verified. Examples of user views, represented and implemented in Prolog and a systems dynamics modelling tool, are presented.

1. Introduction

In recent years, few areas of organizational research have received as much attention as business process modelling, but industry experience with process improvement initiatives has been disappointing. For example, in a 1994 survey of 350 US companies that had undertaken business process reengineering exercises, only 17% of the companies surveyed expressed satisfaction with the results [1]. Sachs [21] has put forward convincing arguments that many failures have resulted from too narrow a focus and that a more holistic view is required. In particular, she argues that "*business process reengineering --- fails to recognise how work is carried out and the part that human ingenuity plays in it*" and that "(1) if only the organizational (explicit structures) of work are considered in designing work and (2) the importance

of learning is left out, there will be negative consequences in the conception and implementation of [process] design".

Software engineering process modelling is a special case of business process modelling and Curtis, Krasner and Iscoe [4] made a similar point when they noted that most software engineering process modelling work has focused on the evolutionary behaviour of artefacts through the various developmental stages rather than on the behaviour of those creating the artefacts. In 1995, Yourdon, Constantine and Thomsett [25] presented an industry seminar in which they stressed that any future, substantial, software engineering productivity improvements are heavily contingent on much greater attention being paid to the so-called "softer" organizational, social and human factors. On the surface, this might suggest that little has been achieved since Curtis et al. [4] flagged the importance of organizational behaviour aspects in their 1988 work - yet, there have been some encouraging developments; notably, the development of the "People Capability Maturity Model" [6]. In addition, the issue of organizational behaviour in software engineering appears to be receiving increasing attention among researchers; e.g. a perusal of the proceedings of the 1996 Americas Conference on Information Systems reveals many interesting papers relevant to this research area (see e.g. [7,22,23]). Nevertheless, few researchers in the area are making any claims that we are much closer today to solving the people-related problems that have remained a constant (and highly significant) dilemma to practitioners since the inception of large-scale software engineering projects some 30 years ago.

Here, a process modelling framework is briefly outlined and its utility is illustrated via examples. A feature of the framework is that it permits the development of more formal models than are commonly found in the organization and management theory (OMT) literature. The case for greater formality in OMT has been argued recently by Jaques [12] who notes that, despite massive

efforts by organization and social science experts, only the merest beginnings of an organization and management science have become evident. In arguing this case, he reserves particular criticism for *"the pile of vague and ill-defined terms that litter the field"* and notes that *"Without -- clear meaning it is impossible to think, or to test propositions, or to talk to one another with any hope of understanding"* (p. 10). These observations are consistent with an increasing body of OMT research which has addressed the benefits of formal models as opposed to informal, literary theorising. For example, Bendor and Moe [2] claim that more formal representations clarify chains of reasoning and simplify the task of verifying that conclusions do indeed follow from assumptions; McGrath [13] points to the clarification of concept overlaps, ambiguities and inconsistencies; and Curtis, Kellner and Over [5] suggest that the properties of multiple model perspectives can be analysed more accurately where representation constructs are formally constrained. In addition, formal models can generally be implemented more readily and modelling both "soft" and "hard" data using a common approach reduces the possibility that the critical softer aspects will be overlooked.

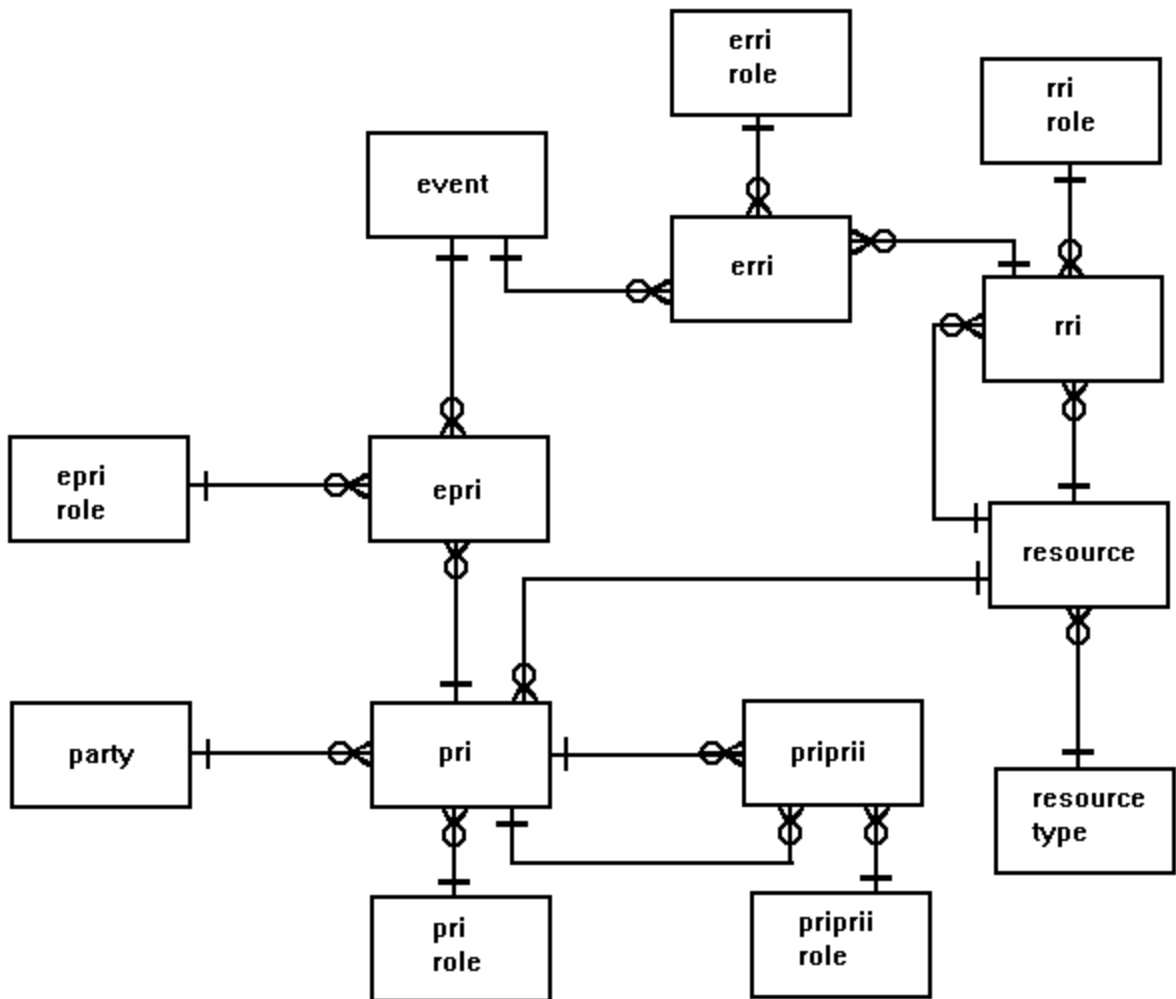
A feature of the framework presented here is that user views, in a variety of modelling formalisms, may be extracted from a common, highly-abstracted conceptual model. This is consistent with a growing consensus that development of software engineering process models requires a "horses for courses" approach. That is, different theories are often best represented using different paradigms, with the conceptual model serving as a common reference point through which consistent external views may be derived and verified. The effectiveness of the modelling framework has been demonstrated in the field through mappings of organizational power, resistance and motivation concepts into external (or user) models implemented in Prolog, a frame-based expert system shell and a systems dynamics tool. Among other applications, the computerised implementations of our models have been used to predict possible sources of resistance to implementations of information systems, information systems' strategies and components of BPR exercises [13]. Here, mappings of organizational power concepts into external (or user) views implemented in Prolog and a systems dynamics package are presented. This work extends previous research into the derivation and automated implementation of formal, AI-based models of OMT [16].

2. A Process Modelling Framework

The framework has three levels: with the universe of discourse (UoD) at Level 1; the conceptual model at Level 2; and the external models (or user views) at Level 3. In developing this framework the work of ISO Technical Committee 97, in defining the foundations of the 3-schema database management system architecture [10], has been drawn upon. Thus the *UoD* refers to *that collection of objects, from a real or postulated world, that is being described*. It should be noted that, while the world of interest here is centred on software design and development processes (and, more particularly, on behavioural aspects of these processes), the framework is applicable to organizational processes in general.

The UoD representation is based on the layered behavioral model of Curtis et al. [4]. The *tech/admin* layer at its core refers to the parties, artefacts, roles etc. that form the basis of most process models; outside that, the *individual* layer refers to cognitive aspects (attitudes, opinions, beliefs, knowledge etc.); next, the *project team* layer focuses mainly on group dynamics; the *company* layer is concerned with corporate-wide aspects, such as organizational power, politics, culture and structure; and, finally, the *business milieu* layer refers to an organization's interaction with its external environment.

At the second level, the *conceptual model* defines *the objects of the UoD, including rules governing allowable classifications, states, transitions and constraints* [10]. Part of the conceptual model is illustrated in Figure 1. The model is represented in entity-relationship form, it is highly abstracted and it is a *common denominator* schema, as defined by Curtis et al. [5]. That is, it is a schema that specifies only the essentials of the vital or core processes, leaving aside any external representation considerations. The entity-relationship approach has been employed not because it is unarguably superior to competing formalisms but because: 1) it is probably the most popular and best known data modelling approach used within the information systems arena; 2) there is a well-defined abstraction process for entity-relationship models that employs much the same "super" entity types used within process modelling; and 3) the whole rationale for the development of the entity-relationship modelling approach was to provide a means of unifying competing (information) modelling formalisms [3]. Data at the conceptual level is represented predominately in relational form, as illustrated in Figure 2 (the tables here contain data used in later examples).



Glossary:

pri	party-resource involvement
epri	event-party-resource involvement
priprii	pri-pri involvement
rri	resource-resource involvement
erri	event-resource-resource involvement

Figure 1: A Simplified, Abstract Conceptual Process Model using "Information Engineering" Conventions [8].

At Level 3, *external models* or *user views* are mappings from all or part of a conceptual model to a language or representational form of a user's choosing [10]. Examples of two such mappings will now be introduced. The examples are concerned with

organizational change and, more specifically, with power source and influence redistributions brought about by the development and implementation of new information systems.

pri	pri-id	party-id	res-id	pri-role
1		Engineering	OPOS	owner
2		Finance	IPOS	sponsor

rri	rri-id	res-id	res-id	rri-role
1	Influence-change	IPOS	OPOS	replacement-for
	Replacement	Replacement	null	g1
	Net-fn-change	Common-ownership	Net-fn-change	*
	Redundancy	New-system	Redundancy	-
	Coverage	Exist-system	Coverage	*
	New-system	Coverage	null	null
	Exist-system	New-system	null	null
	Common-ownership	Exist-system	null	null
		Common-ownership	null	null

epri	epri-id	event-id	party-id	res-id	epri-role
1		IPOS	Engineering	OPOS	threatens

event	e-id	initial-state	final-state	impact
	IPOS	OPOS	IPOS	replacement-for
	Replacement	Sys-ownership _{t-1}	Sys-ownership _t	+
	Influence-change	Influence _{t-1}	Influence _t	+

Figure 2: Partial Relational View of Conceptual Model and Sample Data.

3. Example: Prolog View

This external view example is derived from a field test of our work with a large Australian company and from the considerable body of research into organizational power and resistance to change (particularly, the work of Pfeffer [19,20] and Markus, [17]). The field test initially involved a 3-year case study into the development and implementation of a data-centred information systems strategy. During the study, the author worked closely with the team responsible for strategy implementation and built up a case study database (related mainly to resistance and conflict) from interviews, discussions, observations and various forms of documentation. Details of the study were reported originally in [15] and, more recently, by McGrath [14]. The example concerns a proposal to develop an "Integrated Product Orders System" (IPOS) as a replacement for an existing system

(OPOS). The organization structure of the company involved was very complex. Thus, in our example, we have attempted to simplify matters by using (generic) "Finance" and "Engineering" departments as IPOS sponsor and OPOS owner respectively.

Relation intensions can serve as templates for Prolog assertions. Rarely, however, does this result in Prolog programs with clear declarative interpretations - thus negating one of the language's most important features. An alternative approach is illustrated in Figure 3. This approach relies on the fact that, particularly where abstraction is employed, entity-relationship models tend to be composed of collections of basic constructs of the one type and, further, that these structures are often compounded into superstructures which, again, are essentially of the same type.

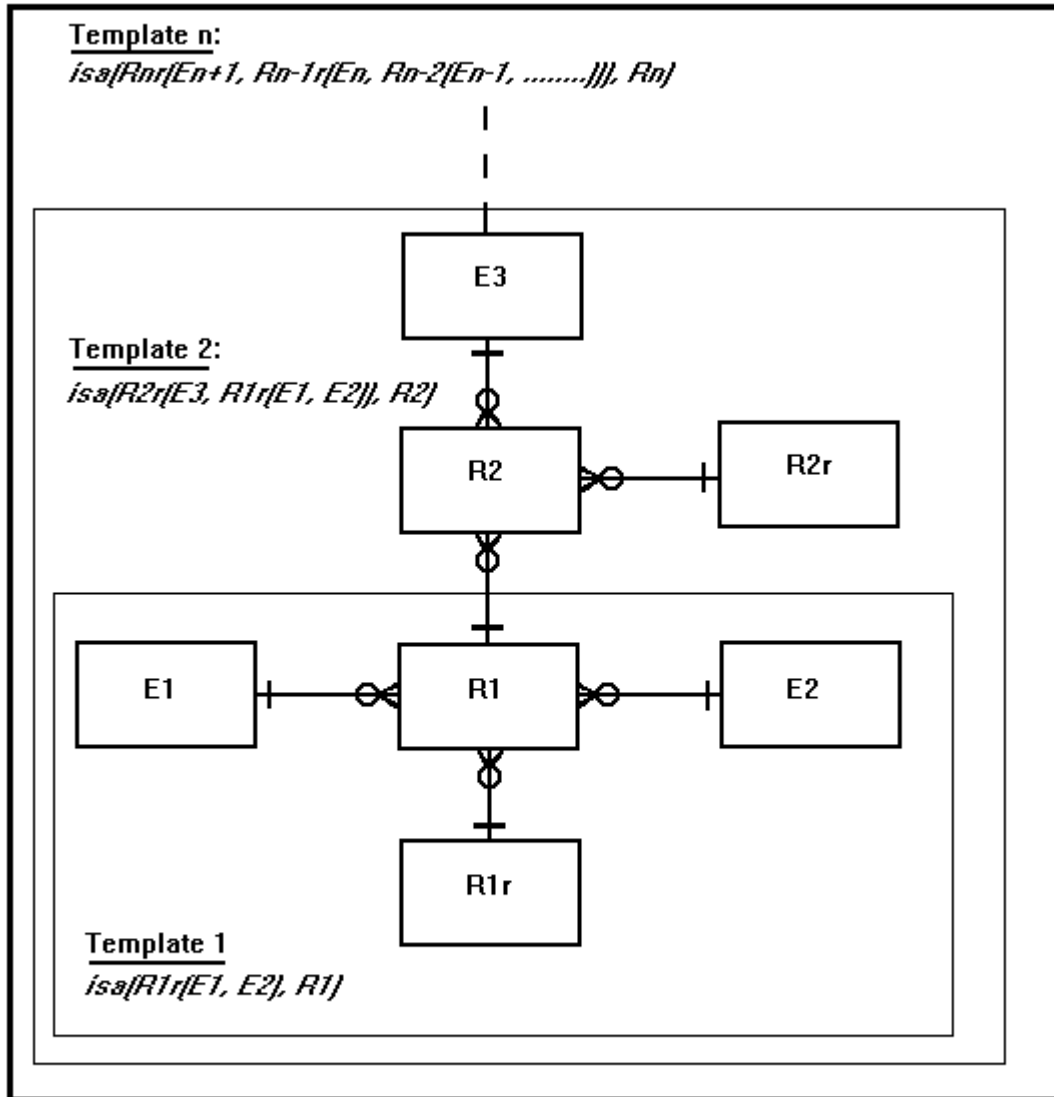


Figure 3: Entity-Relationship to Prolog Mapping Scheme.

An entity, by itself, can be represented simply as an instance of the assertion template¹:

$isa(EI, 'EI')$

where 'EI' is an entity type and EI an instance of an entity of that type. The common construct referred to above is shown at the core of Figure 3. It depicts two entities² E_1

and E_2 , involved in a relationship, R_1 , with R_{1r} signifying the role the first entity plays in the relationship. This construct can be represented as the Prolog assertion template, Template 1:

$isa(R_{1r}(E_1, E_2), 'R_1')$.

Extending now to the next layer in Figure 3, we have a compound structure which is of the same basic type as our core structure. To create the assertion template for this

and relationship types simply as entities and relationships respectively.

¹ As with the de facto Edinburgh standard Prolog syntax, variables begin with capitals or underscores.

² For the sake of brevity, and where there is no possibility of confusion, we shall henceforth refer to entity

structure all that is required is: 1) to replace E_1 , R_{1r} and ' R_1 ' in Template 1 by E_3 , R_{2r} and ' R_2 ' respectively; and 2) to replace E_2 with Template 1's complete subject, $R_{1r}(E_1, E_2)$. Thus Template 2 is:

$isa(R_{2r}(E_3, R_{1r}(E_1, E_2)), 'R_2')$.

This process can be extended indefinitely although, only very rarely would one need to move much beyond two levels. The reader should also note that mapping may begin at different points in the entity-relationship diagram and, at any point in the mapping process, the model can often be extended in a number of directions. As a result, there may be multiple assertion templates at each level.

Returning now to the conceptual process model in Figure 1, one starting point is at *pri*. This results in the assertion template:

$isa(Pri_role(Party, Resource), pri)$

instances of which are:

$isa(owner('Engineering', 'OPOS')pri)$
and
 $isa(sponsor('Finance', 'IPOS')pri)$

which have the declarative interpretations: 1) *(the party) Engineering is the owner of (the system) OPOS is a party-resource involvement* and 2) *(the party) Finance is the sponsor of (the system) IPOS is a party-resource involvement*.

OPOS and *IPOS* are both resources and are related by the fact that the former system is to be replaced by the latter. Thus, using *rri* as a new starting point leads to the new assertion:

$isa(replacement_for('IPOS', 'OPOS'))$.

Further, *IPOS* may also be viewed as an *event*, resulting in the assertion:

$isa('IPOS', event)$.

Pfeffer [19: p.7] defines power as "*a force, a store of potential influence through which events can be affected*", while politics "*involves those activities or behaviours through which power is developed and used within organizational settings*". He describes power as "*a property of the system at rest*" and politics as "*the study of power in action*". Pfeffer's stores of influence are *power*

sources (examples of which are control over information flows, position in the communications network and expert knowledge). Many organization decisions may result in a perceived and/or real redistribution of power sources. There are winners and losers, and losers may resist change.

Thus, resistance may eventuate where an organizational activity poses a potential *threat* to a party's current involvement with a *resource*. Thus, returning now to our original starting point (*pri*), assertions at the next layer conform to the template:

$isa(Epri_role(Event, Pri_role(Party, Resource)), epri)$

and may be derived through the rule:

$isa(threatens(E1, Pri_role(P1, R1)), epri) :-$
 $isa(E1, event),$
 $isa(replacement_for(E1, R1), rri),$
 $isa(Pri_role(P1, R1), pri),$
 $isa(sponsors(P2, E1), pri),$
 $not(P1 = P2).$

Effectively, this rule can be employed to derive details of all threats to parties (involved in *OPOS* in some role) resulting from the proposal to replace that system with *IPOS* (including the threat to *Engineering* retaining its current role as owner of the existing system).

The conceptual view is represented (predominately) in relational form and sample data corresponding to this external Prolog view might be expected to look something like that presented in the foremost rows of the *pri*, *rri* and *epri* tables of Figure 2. The rules for mapping higher-order relations in the Prolog view to their normalised, conceptual representation are relatively trivial. Note also that inferred data is represented explicitly at the conceptual level.

4. Example: System Dynamics View

Within the OMT literature, so-called *messy* problems are commonly characterised by complexity, uncertainty, interrelated sub-problems, recursive dependencies and, crucially, multiple interpretations of the problems' essence. Not unnaturally, such problems are difficult to solve, resolution is generally accompanied by high levels of conflict, and the end result is more often determined by the involved parties' power sources than by rational or (well-founded) bureaucratic processes. In her important work on the role of power and politics in information

systems implementation, Markus [17] has graphically illustrated that many of the problems typically confronted by software engineering project managers are of the messy type.

Vennix [24] has neatly summarised a number of key factors that impede the solution of messy problems. These include: 1) the tendency for each of us to create our own "social reality" based on our background (e.g. functional area, nationality or socio-economic grouping); 2) selective perception and the reinforcement of our (limited) views through recursive loops, often leading to self-fulfilling prophecies; 3) limitations on our cognitive powers, leading us to grossly oversimplify or circumscribe complex problems; 4) a tendency **not** to learn from experience where outcomes are unfavourable (e.g. we may prefer to attribute a failure to bad luck or some other factor outside our control rather than poor judgement); 5) an inability to comprehend multiple, related feedback loops; and 6) multiple problem interpretations resulting from tacit (and, thus, unspoken) assumptions. The focus of Vennix's work is on organizational problem-solving and learning, but his analysis would appear to be particularly apposite to many situations typically encountered during software

engineering activities. A systems dynamics approach has the potential to overcome many of these problems and, in addition: to enhance understanding of the problem under consideration; to foster consensus; and, finally, to create a climate that may lead to increased acceptance of, and commitment to, whatever decision is taken.

Ownership of information systems is a major source of power within organizations [17,19]. Users of systems are heavily dependent on owners (e.g. for maintenance and enhancement needs), owners may direct and control information flows and, finally, they often have significant influence on wider decision making processes (since very few major organizational decisions have no information systems ramifications). Assume now that we wish to assess organizational influence levels resulting from systems' ownership. The newer systems dynamics tools seem well-suited to this task and one such tool, *ithink*TM, has been employed to prepare the external model presented in Figure 4. A brief description of this model is now presented. The reader seeking additional information on constructs and the general modelling approach is referred to [11].

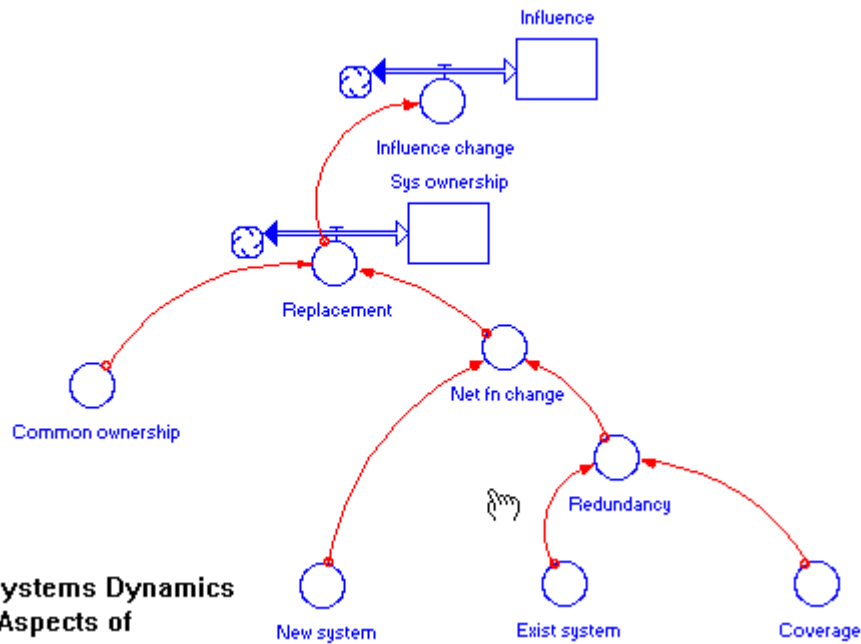


Figure 4: Systems Dynamics Model of Aspects of Organisational Power.

The basic building blocks of systems dynamics models are *stocks*, *flows*, and *converters*. Users of *ithink*TM (and similar products) develop much of their models visually. Little mathematical sophistication is required of the user, since the system provides considerable automated guidance in creating the difference equations [9] that

underpin systems dynamics models. Referring again to Figure 4, *Sys_ownership* and *Influence* are examples of stocks, *Replacement* and *Influence_change* are flows and *New_system* and *Exist_system* (and all other constructs represented as circles) are converters.

The connection between systems ownership and organizational influence in Figure 4 is represented by the arrows linking the flows associated with the two stocks. *Sys_ownership* is measured in terms of organizational functions covered by systems owned (e.g. in function points or similar). This measure will rise and fall as new systems are developed and old systems are replaced (with influence varying correspondingly). Thus, a system *Replacement* can result in a gain or loss of *Sys_ownership*. *Coverage* (a value between 0 and 1) refers to the extent to which a proposed system's functionality overlaps functions covered by a current system. *New_system* and *Exist_system* are measures of parties' (ownership) involvements in proposed and current systems respectively. Thus, the *Net_fn_change* for any party ($= \text{New_system} - \text{Exist_system} * \text{Coverage}$) may be positive, negative or zero (i.e. not involved). *Common_ownership* (like *Coverage*, a value between 0 and 1) allows the impact of other systems covering the functions involved to be factored into the change impact.

Each stock, flow and converter has a single primary measure associated with it. Stocks and flows are *resource* and *event* (instances) respectively. An event involves a resource change of state (from time $t-1$ to time t), with a magnitude equivalent to the event identifier, and a change direction (+ or -) being determined by its *impact*. The equation that is used to determine the magnitude of the change is:

$$R_t = R_{t-1} +/- E_t$$

where R represents a resource and E an event. The event identifier may refer to an *rri* instance. Thus, values for *Sys_ownership* and *Replacement* in Figure 4 are computed (during simulation runs) through the equations:

$$\begin{aligned} \text{Sys_ownership}_t &= \text{Sys_ownership}_{t-1} + \text{Replacement}_t \\ \text{and} \\ \text{Replacement}_t &= *(\text{Common_ownership}_t, \\ &\quad \text{Net_fn_change}_t). \end{aligned}$$

For an indication of how details of this type of external view are stored at the conceptual level, the reader should consult Figure 2 (particularly, the *event* and *rri* tables). Note that flows can be represented as converters which, in turn, can be represented conceptually as *rri* instances. Converters serve several purposes: specifically, they can be used to provide input values to the system, to convert primary measures of stocks or converters to alternatives, or to relate stocks or converters to each other (through arithmetic operators or graphical functions).

The simplest instance of a converter is:

$$C_1 = Op_1(R_1, _)$$

where R_1 is a stock or (input) converter and Op_1 is the null operator. An indefinite sequence of further operations can be applied to these base-level converters, so that the general case has the form:

$$C_n = Op_n(C_{n-1}, R_n).$$

Thus, the converter, *Net_fn_change*, is represented in the conceptual model as the sequence of *rri* relationship instances contained in rows 4-8 and is computed at the external level through the equations:

$$\begin{aligned} \text{Exist_system} &= \text{null}(\text{Exist_system}, _); \\ \text{New_system} &= \text{null}(\text{New_system}, _); \\ \text{Coverage} &= \text{null}(\text{Coverage}, _); \\ \text{Redundancy} &= *(\text{Exist_system}, \text{Coverage}); \text{ and} \\ \text{Net_fn_change} &= -(\text{New_system}, \text{Redundancy}). \end{aligned}$$

5. Sharing Information Between Views

Curtis et al. [5: p.85] have argued strongly in favour of the type of multi-paradigm, external view approach presented in this paper. In particular, they contend that:

"The diversity in audiences, in information content, and in the different descriptive capabilities needed for human vs. machine interaction of processes all impose broad, and sometimes conflicting, requirements on a modelling approach. An approach that integrates multiple representational paradigms is --- considered necessary for effective software process modelling".

At the same time, however, they highlight consistency and information sharing as two of the more significant obstacles in applying this process modelling approach. The framework presented in this paper goes some way towards addressing these problems.

For example, our conceptual model (as represented in Figure 2) contains the results of mappings from the two external views presented in the two previous sections. Assume now that we wish to produce an external model concerned with influence and, in particular, the identification of organization parties that may lose influence as a consequence of threats to their current

system roles. Again, we might employ Prolog as the process modelling formalism, and a starting point for our new external view might be the single rule:

```
isa(may_lose_influence(R1, _(P1, R1)), epri) :-  
    isa_('Replacement', null, rri),  
    isa(replacement_for(R1, R2), rri),  
    isa_(P1, R2), pri).
```

Alternatively, we could construct an additional external view using relational DBMS technology and achieve the same result through the SQL query:

```
SELECT PARTY-ID FROM PRI, EVENT  
WHERE PRI.RES-ID = EVENT.INITIAL STATE  
AND IMPACT = 'REPLACEMENT-FOR'  
AND PRI-ROLE = 'OWNER'
```

Limiting ourselves to the data in Figure 2, the above rule and query, when activated, will derive the fact that *Engineering* may lose influence because their current *OPOS* ownership role is threatened by *IPOS*. This too will be recorded in the conceptual view (as an *epri* tuple) but the result, in itself, is not all that remarkable (but could be extremely important if, e.g., the *IPOS* sponsor was concerned with devising tactics to counter sources of potential resistance to implementation of the new system). The more important aspect is that multi-paradigm, external view data sharing has been realized - since the rule conditions (and relational tuples) are a combination of knowledge derived during the development of the (original) Prolog and systems dynamics views discussed earlier.

6. Summary

The ever expanding range of computing and communications technologies plays only an enabling role in the development of the systems eventually used to achieve a user's defined business objectives. The building of information systems involves, at its core, the still emerging discipline of software engineering as well as other more traditional areas of engineering and science and a number of non-technical management, social and organizational disciplines. In essence, systems are not built in a vacuum, but within organizational environments where outcomes are heavily influenced by a myriad variety of internal and external socio-technical factors. Software design is essentially a problem-solving activity, yet few software development models have taken advantage of the rich body of empirical research on design problem-solving reported in the social science literature.

In addition, tools and methodologies, designed to assist individual activities, often do not scale up because of team and organizational factors that impact on larger projects. Softer issues should be given the same weight in information systems development and implementation processes as the more technical features. Furthermore, to the extent possible, both soft and hard aspects should be modelled using the same set of paradigms. A framework has been presented as a vehicle for incorporating vital socio-technical and behavioural aspects (taken from the OMT literature) into software engineering process models. The aim is to empower project managers and others involved in systems development to better deal with the critical human aspects inherent in all non-trivial projects.

Early OMT models were highly quantitative: principally because, as noted by Masuch [18], if a computerised implementation was desired, model developers had little alternative. Masuch though, further notes that much OMT is highly-qualitative and this, combined with the relatively recent proliferation of powerful AI tools outside laboratories, has resulted in a pronounced shift towards qualitative models. Despite this trend, however, he cautions against over-enthusiasm for the qualitative approach and quite sensibly suggests that a quantitative conceptualisation scheme should always be employed where details of the phenomenon under study are most naturally expressed in quantitative terms. This accords with the modelling philosophy underpinning this work where, despite employing a common approach at the highly-abstracted conceptual level, user views (mappings from the conceptual model) may be essentially qualitative, quantitative or a mixture of both. This is consistent with the views of an increasing body of process modelling researchers who have argued that different modelling objectives, user diversity, conflicting requirements and the need for both large and small-grained levels of abstraction all demand OMT models permitting multi-paradigm representations.

References

1. "It's time to re-engineer the engineers", *The Australian Newspaper (Computers and High Technology section)*, (reprinted from *The Economist*), 12 July, 1994.
2. Bendor, J. and Moe, T.M., "Bureaucracy and Subgovernments - A Simulation Model", *Artificial Intelligence in Organization and Management Theory*, (M. Masuch and M. Warglien eds.), North-Holland, Amsterdam, 1992, pp. 119-141.

3. Chen, P.P.S., "The Entity-Relationship Model - Towards a Unified View of Data", *ACM Transactions on Database Systems*, Vol.1, No.1, 1976, pp. 9-36.
4. Curtis, B., Krasner, H. and Iscoe, N., "A Field Study of the Software Design Process for Large Systems", *Communications of the ACM*, Vol.31, No.11, 1988, pp. 1268-1287.
5. Curtis, B., Kellner, M.I. and Over, J., "Process Modelling", *Communications of the ACM*, Vol.35, No.9, 1992, pp. 75-90.
6. Curtis, B., Hefley, W.E. and Miller, S., *People Capability Maturity Model*, (CMU/SEI-95-MM-02), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1995.
7. Epstein, M., "The Role and Worldview of Systems Designers: A Multimethod Study of Information Systems Practitioners in the Public Sector", *Proceedings of the Americas Conference on Information Systems*, Phoenix, Arizona, 1996, pp. 55-56.
8. Finkelstein, C., *An Introduction to Information Engineering: From Strategic Planning to Information Systems*, Addison-Wesley, New York, 1989.
9. Forrester, J.W., *Industrial Dynamics*, MIT Press, Cambridge, Massachusetts, 1961.
10. van Griethuysen, J.J., *Concepts and Terminology for the Conceptual Schema and the Information Base*, ISO Technical Report ISO/TC97/SC5/WG3, 1982.
11. Getting Started with ithink: *A Hands-On Experience*, High Performance Systems Inc., Hanover, NH, 1994.
12. Jaques, E., *Requisite Organization*, (final draft, revised 2nd edition), Cason Hall, Arlington, VA, 1996.
13. McGrath, G.M., "MP/L1: Towards an Automated Model of Organisational Power", *Proceedings of the 2nd IEEE Australian and New Zealand Conference on Intelligent Information Systems*, Brisbane, Australia, 29 Nov. - 2 Dec., 1994, pp. 487-491.
14. McGrath, G.M., "An Implementation of a Data-Centred Information Systems Strategy: A Power/Political Perspective", *Journal of Failure and Lessons Learned in Information Technology Management*, Vol.1, No.1, 1997, pp.3-17.
15. McGrath, G.M., Dampney, C.N.G. and More E., "Yet Another Information Systems Strategy Implementation Failure: An Exploratory Case Study", *Proceedings of the 5th Australasian Conference on Information Systems*, Monash University, Melbourne, Sept. 27-29, 1994, pp. 649-660.
16. McGrath, G.M., Dampney, C.N.G. and More, E., "MP/L1: An Automated Model of Organisational Power and its Application as a Conflict Prediction Aid in Information Systems Strategy Implementation", *Proceedings of the 11th IEEE Conference on Artificial Intelligence for Applications*, Los Angeles, Feb. 20-22, 1995, pp. 56-64.
17. Markus, M.L., "Power, Politics and MIS Implementation", *Communications of the ACM*, Vol.26, No.6, 1983, pp. 430-444.
18. Masuch, M., "Introduction - Artificial Intelligence in Organisation and Management Theory", *Artificial Intelligence in Organization and Management Theory*, (M. Masuch and M. Warglien eds.), North-Holland, Amsterdam, 1992, pp. 1-19.
19. Pfeffer, J., *Power in Organizations*, Pitman Pub. Inc., Marshfield, Massachusetts, 1981.
20. Pfeffer, J., *Managing with Power: Politics and Influence in Organisations*, Harvard Business School Press, Boston, Massachusetts, 1992.
21. Sachs, P., "Transforming Work: Collaboration, Learning and Design", *Communications of the ACM*, Vol.38, No.9, 1995, pp. 36-44.
22. Sethi, V., Meinert, D., King, R.C. and Sethi, V., "The Multidimensional Nature of Organizational Commitment among Information Systems Personnel", *Proceedings of the Americas Conference on Information Systems*, Phoenix, Arizona, 1996, pp. 52-54.
23. Subramani, M. and Henderson, J., "The Influence of Convergence on IS Service Quality", *Proceedings of the Americas Conference on Information Systems*, Phoenix, Arizona, 1996, pp. 57-59.
24. Vennix, J.A.M., *Group Model Building: Facilitating Team Learning Using System Dynamics*, Wiley, Chichester, UK, 1996.
25. Yourdon, E., Constantine, L. and Thomsett, R., *Reinventing IS/IT*, Australian Computer Society, Seminar Proceedings, Sydney, Australia, May 24, 1995.