EMPIRICAL MODELLING FOR BUSINESS PROCESS REENGINEERING: AN EXPERIENCE-BASED APPROACH

Yih-Chang Chen, Steve Russ, Meurig Beynon
Department of Computer Science, University of Warwick
Coventry CV4 7AL, United Kingdom

Telephone: +44 (0)24 7652 3193
Facsimile: +44 (0)24 7657 3024
E-mail: {cheny, sbr, wmb}@dcs.warwick.ac.uk

ABSTRACT

Most business processes rely on informal knowledge and social behaviour but these are areas which have not, so far, been well suited for modelling with computer-based techniques. We describe a new experience-based approach to modelling with computers which has natural application to business process modelling. A framework using this approach, SPORE (situated process of requirements engineering), is extended to encompass applications to participative BPR (i.e. supporting many users in a distributed environment). An outline of an application of our methods to a warehouse management system is included.

Keywords

Business process reengineering, business process modelling, experience, requirements engineering, use case.

1 INTRODUCTION

Modern corporations are faced with a highly competitive market environment changing at an accelerating rate. In the early 1990s several analysts were suggesting that the conventional incremental style of organisational change was inadequate for this challenge. To gain competitive advantage or even maintain market position, it was argued, would require so-called 'radical' change. It had become common to use the metaphor of 'engineering' to describe change that is planned or designed (cf. 'software engineering'). So it was natural for describing this new order of change that 'Business Process Reengineering', or 'Business Process Re-design' (BPR) became the preferred terms. From this perspective it was crucial that business processes should be re-designed in a cross-functional process 'vision' guided by overall objectives and new resources, particularly the resources of IT. There was an optimism for IT reminiscent of the early days of artificial intelligence: "IT capabilities .... can work miracles by the standards of previous generations. How else but through this technology can we manage our processes globally, instantly, efficiently, and correctly? It is clear that no other tools are comparable." (Davenport, 1993) However, it was not to be long before disillusion with the BPR vision appeared. In 1996 Davenport himself published an article entitled, *Why Re-engineering Failed: The Fad that Forgot People* in which he admits:

> To most business people in the United States, re-engineering has become a word that stands for restructuring, lay-offs, and too often, failed change programmes...companies that embraced [re-engineering] as the silver bullet are now looking for ways to re-build the organisation's torn fabric. (Davenport, 1996)

In 1998 it was reported that only around 30% of BPR projects were regarded as a success (Galliers, 1998). The earlier promise of BPR had not been fulfilled. One reaction to this outcome was to retain faith in IT as a dominant support and just admit that since it could not adapt – or at least not at acceptable levels of cost – then business activities must adapt to IT. For example:

> The pendulum has swung from 'continuous reengineering and re-inventing' to 'pick an application package and force our business processes to comply with the package'. (Riemer, 1998)

Another response was to be more relaxed over the likely role of IT in business:

> IT can often be a catalyst in this process [of change] and IT opportunities for new or enhanced products and services should certainly not be overlooked. (Galliers, 1998)

There are, no doubt, many reasons for the limited success of the BPR programme. It was surely over-hyped in the first place. There is only a certain amount, and rate, of change that people and organisations can accommodate while maintaining their basic business objectives. Most business processes depend crucially at every point upon people and their informal knowledge and social behaviour. But these are areas for which conventional computer-based techniques are not well suited and there was, and still remains today, a substantial gap between the need to model business process innovations and the capabilities and mechanisms available from IT to support the task.

In so far as IT is itself the problem here – as opposed to the solution it was intended to be – the problem lies more with software than with hardware. Hardware developments – multimedia functions, networks, storage and processor performance, screen display – have been impressive over recent years. But although object-oriented methods have made an important contribution, the software 'crisis' has still not been solved. Taking proper account of human factors is well known to be a major challenge for all interactive software. And the first human factor to be considered is the requirement of the software system. The most sensitive and difficult area of

software development lies in requirements engineering. Should this be a phase with an end-point – as the programmer would prefer? Or should it be a continuous evolution – as the customer would prefer? There are notorious difficulties for conventional development methods in having to define, and make a commitment to, a hard system boundary in advance of any system development work. For example, a requirement specification is often referred to as the basis for a contract between the developer and the customer. That specification, and contract, represents the documentation of such a boundary. At the design level the concept of a rigid boundary re-appears in making decisions about the objects or components to be used in an application program. The difficulties that affect object-oriented methods in analysis and design are discussed in (Kaindl,1999) with reference to the movement from the problem domain (real world application) to the solution domain (world of programs and systems).

In this paper, we introduce a novel, human-centred approach to modelling and system development. It is human-centred in taking seriously the subjective experience of the modeller – both as a starting point for model construction and as a guiding principle throughout development. The central role in our approach given to observation and experiment has led to it being called 'Empirical Modelling' (EM). It is not so much a methodology as a broad outlook on computing which has far-reaching consequences. Indeed it can be thought of as a reengineering of the computing process itself. We describe it in greater detail in section 3. Then in section 4 we explain how EM can be applied to BPR. The main idea here is that it is essential within EM to take account of the wider context of a desired 'system' in terms of the purposes, people and other resources which will form the environment of the system. A problem-oriented framework SPORE (situated process of requirements engineering) is described. By applying SPORE to the software requirements of a business, possible solutions to problems in the business domain can be explored in an open-ended and situated manner. We propose that the SPORE concept can be extended to support effective and efficient participative BPR. Within this framework, people participating in the business process can create and use the models as a powerful means of supporting their collaborative interaction for 'growing' solutions, or reengineering processes, in a distributed environment.

In section 5 a case study applying this framework to a warehouse distribution system is discussed. A 'use case driven' version of this case study appears in (Jacobson et al., 1992). In contrast to the Jacobson version we model some of the processes as they might have been prior to the proposed computer system. They are modelled as a series of interactions between agents (both human and non-human agents). The resulting environment is suited to reengineering the processes through negotiation between the existing (problem) situation and possible solutions defined by requirements for system components (including software).

2 BUSINESS PROCESS REENGINEERING

We sketch here an outline of some of the issues and problems involved in BPR. Although this discussion is brief, we hope it will be sufficient to show the relevance and potential of the approach (EM) that we describe in later sections.

2.1 The Key Concepts

In 1993 two key publications (one by Hammer and Champy, another by Davenport) brought widespread attention to the emerging field of BPR. The very concept of 'business process' required a re-orientation of managers' thinking about their business activities. It is defined by Hammer and Champy (1993) as 'a set of activities which produces an output valuable to the customer'. This concept cut across traditional boundaries in the structure of a business (e.g. departments, sections, functions). For example, the process of new product development cut across departments for R&D, for manufacturing and for marketing. Then 'reengineering' meant discovering how a process currently operates, re-designing that process to improve efficiency and remove wastage, and finally implementing the new process using whatever enabling technology was appropriate. The force of the new terminology was to draw attention to a perceived need for 'radical change', not mere 'improvement'. The scale of change envisaged by the term 'reengineering', or 'innovation' as preferred by Davenport, is described as follows:

> Objectives of 5% or 10% improvement in all business processes each year must give way to efforts to achieve 50%, 100%, or even higher improvement levels in a few key processes. (Davenport, 1993)

The lesson we take from this is that when the business context and resources are changing rapidly, radical change may need to take place regularly. In order to model such changes effectively with computers we require environments with the greatest flexibility.

2.2 Participative BPR

BPR seeks to devise new ways of organising tasks, organising people and making use of IT systems so that the resulting processes will better support the goals of the organisation. Vidgen et al. (1994) define the central tenets of BPR as:

- radical change and assumption challenge;
- process and goal orientation;
- organisational re-structuring;
- the exploitation of enabling technologies, particularly information technology.

Thus, BPR has more of an organisational focus than a technical one. The effort is directed at changing people's thinking and must therefore take into account expectations and viewpoints. The process view of the business activities involved in new product development, for example, reflects the customer's viewpoint more than the producer's viewpoint.

While the thought of the scale of change mentioned above by Davenport might be intoxicating for managers

to consider, it is likely to be a less exciting prospect for those employees who are directly driving such order of magnitude changes in performance. Thus Sherwood-Smith (1994) advocates a less strident form of BPR which is people-centred and driven by needs, rather than by IT.

> Administrative systems involving people should not be reengineered, they should be participatively re-designed. (Sherwood-Smith, 1994)

Such a participative approach respects the culture and social context of an organisation. This demands a high degree of communication and evaluation. In reference to CASE tools supporting BPR, Sherwood-Smith continues (in the same paper):

> Because we believe Business Process Re-Design is essentially a group activity and should be participative, one key aspect of the tool set is that it must run in a collaborative environment.

This sentiment is an exception to what seems to be the more usual undemanding and uncritical attitude to IT from authors on BPR. Available software resources are often accepted as given and their limitations go uncommented although it is, we suspect, precisely their profound limitations that are a significant factor in the 'failure' of some BPR efforts. Many current applications are designed in a 'take it or leave it' fashion which is inappropriate to a rapidly changing business environment. With frequent mergers and outsourcing of activities businesses need computing environments which support unforeseen changes in needs and can exploit opportunities as they arise. The EM approach creates such environments and they support collaborative working.

2.3 Modelling Business Processes

Organisations and their business processes are complex. Understanding anything involves making some kind of model of the thing in our heads. Thus a fundamental motive for modelling business processes is to help understand them. It is controversial in the BPR literature exactly how much understanding of a process is necessary or desirable prior to its re-design. Hammer and Champy (1993) argue that a very detailed analysis of process is not needed because the goal of the reengineering effort is not to improve the existing process but to design a 'totally new and superior design'. As both Hammer and van Meel et al. have observed most business structures have not been designed at all, but have simply 'emerged'. But such evolution has occurred in a social and technical context that is not arbitrary. There are usually reasons for the way things emerge and sometimes they turn out to be very important reasons (that are only discovered after a disastrous re-design!). So we take the view that in the case of processes of any complexity it is important to have as thorough an understanding of the process as possible. It is only then one can dare responsibly to propose a really new design. Since this is a controversial issue we note that (Jacobson et al., 1995) supports our view. In a chapter devoted to this theme ('Reversing the Existing Business') they write:

> ... we do believe that you need a good picture of your current organization before you can finally decide on the best way to change it. If the reengineers understand the business as it is today, they will be able to avoid making unfeasible change proposals. (p.153)

Modelling a business process helps us to understand it and so to re-design it. In a comprehensive, Germanic style the modelling task has been described as follows:

> Modelling business processes first means to express the flow and the dependencies of steps in the respective processes in order to make the dynamic behaviour explicit, to be able to communicate it, to analyse it with respect to possibilities of improvement, and to use it for simulations as well as for controlling automated workflow. (Schader and Korthaus, 1998)

This statement effectively summarises the four 'requirements' of a modelling method for BPR given by Gerrits (1994). Gerrits emphasises the role of simulation both for assessing the quality of the models of the current situation on the one hand, and the performance of the re-designed processes, on the other hand.

According to van Meel et al. (1994) the methods for achieving business engineering given in the literature 'roughly follow the pattern of general problem solving'. To offer more support for BPR these authors suggest a model-based problem solving approach summarised in Figure 1. The term 'empirical model' used by these authors means something very different from our usage in EM as described in section 3. For example, the van Meel empirical model is a text-based version of a conceptual model; an EM model is a computer-based version of an experiential model. Nevertheless there are striking structural similarities between the van Meel approach and ours, for example in the notion of an experimental, correspondence check between the model and the problem situation. We mention this approach also because the framework here of *problem situation and solution [space]* is a recurring and unifying theme of our approach. It is one that has recently been used in EM work on requirements, and we apply it in this paper for

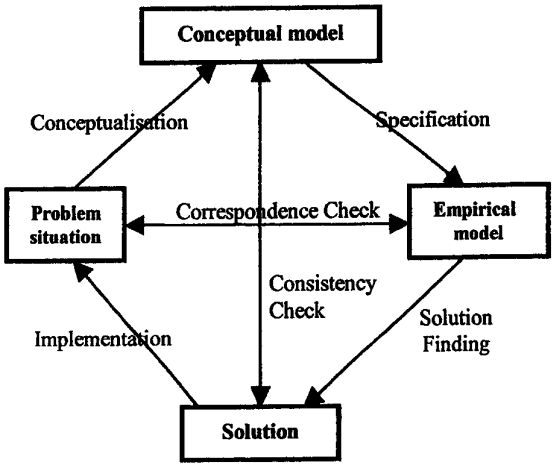

Figure 1. The Process of Problem Solving
(source: amended from van Meel et al., 1994)

the application of EM to BPR.

2.4 Object-Orientation and Use Cases

The emergence of BPR in the early 1990s coincided with a dramatic rise in claims and interest in object-oriented methods of software development. There was unprecedented investment in OO by industry and huge co-operation between industries to establish standards and tools for OO (e.g. the formation of the powerful Object Management Group). The was a mushrooming of textbooks and obligatory courses on OO academia. Since BPR laid emphasis on exploiting the resources of IT, it was natural that those in charge of reengineering projects should turn to OO methods. Among many others the series of well-known works by Jacobson and his co-authors (e.g. Jacobson et al., 1992; Jacobson et al., 1995; Jacobson et al., 1997) no doubt helped to forge these links between BPR and OO approaches to system development. The 'use case driven' development method for software was the subject of the first work, then it was applied in (Jacobson et al., 1995) to the modelling of business processes. A use case is literally a 'case of use' of a proposed or existing system and refers to some specific use by an 'actor' (a role of a potential user of the system). It is a sequence of events corresponding to some function of the system useful to the user. At one level such a sequence or 'course' of events corresponds closely to a (business) process. At a lower level of abstraction it may, in some cases, also correspond to a functional requirement for a software system. The collection of all use cases constitutes the use case model which is a major part of the requirements model in Jacobson's software development method.

Schader and Korthaus (1998) draw attention to the range of attributes (e.g. 'iterative', 'use case driven', 'incremental') which can apply equally to both software development and business modelling. As Warboys et al. (1999) state:

> Both the business process practitioner and the software process modeller have much in common ..... One has to design dynamically changeable and efficient processes, and the other the process knowledgeable software to support them.

Another link between BPR and software development lies in the common pattern of negotiation between a problem situation and a solution space. In BPR, this is reflected in the optimum level of goal satisfaction in the light of resources available, and in software, in the ongoing evolution of a requirement. In a way similar to the work of Jacobson et al. (1995), Nurcan et al. (1998) also try to identify and describe business processes by use cases and scenarios. They describe the relationship between the business process and a use case as:

> A use case specification comprises a description of the context of the BP, the interactions between the agents involved in the BP, the interactions of these agents with an automated system supporting the BP and attached system internal requirements.

It is now widely recognised that the concept of use cases is independent of object-orientation. But both ideas, highly influential as they have been over the past decade, have also not been without their critics. We have already mentioned (Kaindl, 1999) for drawing attention to the deep problems, for long generally unacknowledged, that arise when representing informal objects in an application domain by formal objects in the programming domain. Among others, the series of papers by Simons (e.g. Simons, 1999; Simons and Graham, 1998; Simons and Graham, 1999) describe some of the serious semantic confusions surrounding use cases and object modelling.

We share with conventional methods for system development the aim of harnessing computer power to solve problems and do useful tasks. But we are doubtful that there are universal 'methods' for doing this and doubtful that the required behaviour of a complex 'system' can successfully be prescribed in use cases in advance of the construction and use of the system. This seems to us particularly true of volatile contexts with a high dependency on human factors – such as many business processes. This radical, and perhaps immodest, scepticism is one motivation for building models in the way we do in EM. Our ultimate aim is, indeed, to develop useful systems, but to do so we suggest the need in general to take a more roundabout, indirect route. We need first to understand what will be useful and how this might change. For this we need to model a wider context than that of the future system itself and EM offers the possibility to do this.

3 EMPIRICAL MODELLING: AN OVERVIEW

Empirical Modelling (EM) is an approach to computer-based modelling that has been developed at the University of Warwick since 1983. In this section we will give an overview of EM and summarise how our approach differs from conventional modelling methods.

3.1 Principles

The EM principles are based on the concepts of observation, agency and dependency. The initial analysis of a domain to be modelled is made by identifying *observables* considered relevant by the modeller. Then these observables are grouped around the *agents* regarded as 'centres', or sources, of change in those observables. The other source of change in the observables is where there are *dependencies* holding between them expressed by definitions. These are 'law-like' dependencies like Newton's law, the physical constraints of rigidity in a solid material, or the conventions of when a game has been won. All these identifications (of observables, dependencies, and agents) are provisional and subjective: they represent the viewpoint of the modeller. The dependencies between observables are expressed in *definitions*. A set of definitions – a *definitive script* – corresponds to a single state of the model. Any particular state of the model should directly correspond to a possible state of its external *referent*. EM is an agent-oriented approach. That is, a key idea is to attribute the state-changing activities that do not arise from dependency maintenance to agents which are associated with certain observables. Here an agent can be a human actor or any other state-changing

component or device. The identification of agency and dependency is based on previous knowledge and experience, and supplemented by new observation and experiment.

There is therefore a primary emphasis in EM on state, not in a public sense but *state-as-perceived-by-agent,* or *state-as-experienced.* It is for this reason that we describe our approach as 'experience-based'. This distinguishes EM from many conventional approaches to modelling in which concepts of state are taken for granted and more attention is given to reproducing desired behaviours. The use case driven method mentioned in section 2.4 is an example of this emphasis on behaviour.

3.2 Techniques

The main technical focus of our modelling approach is the so-called *interactive situation model* (ISM). An ISM is open to experiment in much the same way that its real world referent is open to experiment. That is, we can devise changes in the ISM, introduce new factors, and have direct experience of the results in any way we choose at the time. This leads to patterns of interaction, and a quality of close human engagement in the interaction which is unusual in computer-based models with the major exception of spreadsheet models. The concept of an ISM in fact generalises the spreadsheet in several radical ways (Rasmequan et al., 2000). The word 'situation' in ISM refers to the fact that the model is rooted in a concrete context that affects the modeller's expectation and interpretation. The model is partial, but not disconnected from the physical world.

All EM models, and conventional spreadsheet models, are examples of ISMs. These models incorporate the agency and dependency revealed during the analysis and construction processes. At the same time they maintain the correspondence between the values of variables in a definitive script and the values of observables perceived by the modeller in both the model and its referent. So each state of the computer model may be directly perceived alongside the corresponding state of its referent. There is no preconceived systematic process to be followed in analysing a domain and constructing the ISM. That is, the modelling activity is genuinely open-ended, observations may agree or disagree with expectations and thereby affect further construction. An ISM always presents only a partial and provisional artefact to the modeller.

3.3 Notations

The roles of agents are documented using a notation that classifies observables as follows: those whose values act as stimuli for action (*oracles*), those that can be conditionally redefined (*handles*), ones whose existence depends on that of the agent itself (*states*), the relationships between observables representing the interaction between the agent and its environment (*derivates*), and the privileges of agents for state-changing action (*protocols*). We record these observations in an informal but structured notation known as LSD. Such an LSD account reflects the internal

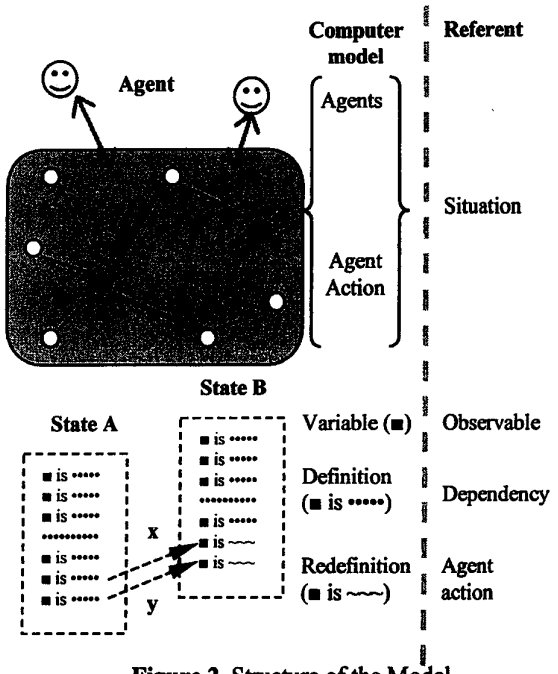

Figure 2. Structure of the Model

perspective of each agent in the account together with the external perspective of the modeller. Such an account is not essential but can be useful throughout the construction process. It can be maintained and refined while the main scripts are being developed.

The definitions, representing dependencies, play a central technical role in EM. A typical definition takes the form of

$$x \text{ is } f(a,b,c)$$

where x, a, b, and c are associated with observables in the referent. The variables a, b and c are defined elsewhere in the script and f is some operation which reflects the perceived dependency of x on a, b and c. The operation f will usually be some user-defined function. Such a definition is a one-way dependency rather than a constraint. It expresses the fact that changes in a, b and c will result in indivisible change in x; the 'is' in the definition is a keyword indicating that this dependency will be maintained automatically. The values of variables represent the observed state from the point of view of the modeller. Changes of state occur either through re-definitions or the addition of new definitions. A script containing many definitions may have hundreds of automatic updates arising from a single re-definition. A simple event or action will typically be represented as a collection of several re-definitions grouped together. The notations providing all these facilities are known collectively as EDEN. They comprise a general purpose definitive notation together with other notations for line drawing and window management. These three notations can be used freely together in the construction of models. There is some limited assistance to users for the management and debugging of scripts but EDEN is primarily a research tool – it supports the principles of

EM but lacks the robustness, efficiency and consistency needed for general end-user development.

3.4 The EM Modelling Process

There is a fundamental difference between EM and conventional modelling in the way the modeller interacts with the state of the model. In a procedural, or object-oriented, language a door object, for example, would be defined with attributes and methods carefully planned in advance of being programmed. These attributes (e.g. dimensions and manner of opening) and methods (e.g. for changing position and for display) form the door *data* for manipulation in the computer model. That data determines in advance the complete state space of the door and all possible interactions with the door. The program can, of course, be edited and re-compiled. But it remains an inherent feature of the paradigm that the program forms a boundary within which state and interaction must be preconceived. The interpretation of program state (even in the presence of visualisation) must take place across this boundary. That is, interpretation involves an association between real-world observations and program abstractions. The validity of this association – which depends on such factors as context and purpose – generally requires human judgement. The computer's manipulation of data is oblivious to such matters and when exceptions and errors arise to render an interpretation invalid, major problems are likely.

Within EM the language for the description of state is the language of *observables*. The state of the model is presented to the user by a perceptual process that is of the same kind as that by which we apprehend state in the real world. There is, then, a comparability and connectedness between observations of the model and observations of its referent. This comparability is lacking in conventional computer modelling because there the 'observation' of the model is the reading of an abstracted value, or the preconceived interpretation of a preprogrammed display. The real-world observation is a subjective, situated experience. The association required for interpreting the model is therefore an association between two things of quite different kinds.

There are three features of an EM model that give the comparability referred to above a special leverage. The link between the observable and a variable in a definition is direct and simple. This contrasts with the situation in many procedural programs where the interpretation of variables and their states can become highly problematic. We acknowledge that the variables in a definitive script must have the 'abstracted' quality we referred to above in connection with conventional programs. But this is a 'limited abstraction' in which the connection between the concrete and the abstract is deliberate and familiar – like that in the use of natural language for description of the world. As described in (Beynon et al., 2000a):

> There is a fundamental mismatch between abstract data that is interpreted by the human in direct association with its counterpart in the real-world referent and situation, and abstract data that is manipulated according to computational rules that can only take account of prespecified and preprogrammed features of this association.

The second feature is that the observation of the model is through a visualisation which is indivisibly linked to the script of the model. The very definition of a line in the line-drawing notation in EDEN is accompanied by its display, just as the existence of a physical edge in the field of vision of a person with normal faculties is automatically accompanied by its perception. The combination of the directness of these two features make for the third feature – the quality of interaction offered in an EM model. There is no preconceived limitation in the revisions the modeller may make at any time in the model and there is a built-in coherence and integrity to all interactions which invoke dependencies. There are some kinds of interaction where there are automated actions from which the human user is excluded. This is appropriate where there are specific and clearly prescribed functions – for example, when using a canned drink dispenser or a washing machine. In such interactions we do *not* wish for more flexibility or more scope for human intervention. But it is when there is no such function to be prescribed – for example, in conversation or driving a vehicle – when the 'purpose' of the interaction is to explore or experiment, that we need a different kind of interaction: one in which the participants can have a close and continuous engagement with each other. It is this latter kind of interaction which – in principle, if not yet always in practice – is offered in an EM model.

Closely allied with the difference just described concerning interaction with state is another fundamental difference between EM and conventional modelling. This is the way change of state takes place. In a conventional program change methods must be planned and preconceived as described above. The control of change is 'handed over' to actions within the program boundary. When these actions are complex and the context alters, or error conditions occur, it may be very hard to make the appropriate changes. In an EM model change either occurs as a dependency update which should always have clear and simple real-world semantics – maintaining the integrity of state – or the change occurs as a direct action of an agent. Initially, except in very simple predictable circumstances, this will be a human agent. Only when certain patterns of change have been experimented with and are known to be operating with reliability over a wide range of local states can those patterns be delegated to an 'automated' agent action.

The agency concept is a much more primitive one than that of control structures in conventional programming. But it is a very general and powerful notion that allows EM to encompass many conventional paradigms for programming and state change. Furthermore, the notion of dependency is a natural way to preserve the integrity of state change, and the real-world semantics of interaction, in a direct and comprehensible fashion.

While observations of the computer model in EM have a qualitative similarity to observations of the world, they are inevitably limited by properties of the interface

(e.g. size of pixels on display) and typically require the use of visual metaphor to supplement direct observation. For the sake of understanding a domain, the faithfulness of experimental interaction matters more than the faithfulness of the representation. For example, in understanding electrical circuits the iconic representation of components, together with selected measurements is preferable to photographic images of components and comprehensive measurements.

Finally in this section we mention three further differences between EM and conventional modelling. Firstly, there is really no counterpart in EM to the 'planning' phase mentioned at the beginning of this section with regard to the example of a door object. This is because such early conceptual modelling in EM can conveniently be directly put into a script with a visualisation and experimented with on the computer. Secondly, the experimentation referred to here, and in relation to the establishment of reliable components mentioned above, corresponds in some measure to the testing of conventional models or programs, but it is significant that this testing occurs here *in advance of any commitment* to a particular form of program. And thirdly a further symptom of the difference in approach of EM is the stage at which we consider an interface to a desired system. It is typical of rapid prototyping approaches to offer a 'mock-up' of an interface to a future system at an early stage. We see an example of this in the use case description given on p.351 of (Jacobson et al., 1992). In an EM development it is typical that the interface is left until an advanced stage of the development – when the purpose and requirement has been clarified through extensive use of the very open-ended phase of model construction and exploration.

4 APPLYING EM TO BPR

4.1 The Wider Context

Having surveyed some of the issues and problems presented by BPR (section 2), and reviewed the broad approach to computation and modelling that is EM (section 3) we now turn to showing how the principles and tools of EM are well suited for application to BPR. Business engineering calls for technical support at two levels that we shall call the *cognitive* and the *operational*. The cognitive level corresponds with the essential processes of understanding the business as a whole, its place in the wider world, and the place of existing processes within the whole. Such understanding is pre-requisite to any reengineering of processes. The operational level refers to the provision of systems which workers can use to perform business activities effectively and efficiently. We envisage that such systems will typically comprise people as well as software components and other kinds of device. Object-oriented business engineering seeks (as EM does) to provide a common framework for both the cognitive and operational levels of support.

There has been a multitude of approaches to implementing BPR and even within the OO approaches Jacobson's is one among many. We used the terminology and concepts of Jacobson's work in section 2.4 above because it is well known and illustrates many of the issues surrounding the use of technology for BPR. In that section use cases are described as corresponding at one level to business processes, and at another level to the functional requirement of software systems. How we understand the relation of business processes to software systems is crucial for understanding the potential contribution of EM to BPR. Part of Jacobson's viewpoint is (Jacobson et al., 1997):

> The object-oriented business engineering models are similar in spirit to those of Object-Oriented Software Engineering (OOSE). The biggest difference is that the 'system' being modelled is now a business organization instead of a software system.

Adopting the viewpoint of EM we would claim that modelling a business organisation and modelling a software system are not so very different. But we are approaching the difference that does exist from the opposite direction to that of an OO approach. Our principles and tools very directly support the modelling of a collection of agents performing structured activities in an open environment where unforeseen changes may occur at any time. This is a description which maps onto a business environment more naturally than that of a software system. It is also plausible that many software systems can be viewed as circumscribed special cases of business processes. Thus when approaching the application of EM to BPR we are not immediately confronted with the usual mismatch that arises when the informal problem world (of business) meets the formal solution world (of programming).

In OOSE, the collection of all use cases, each associated with an actor, forms the use case model. Such modelling is the major technique used in the early stages of Jacobson's approach to offer cognitive support to business engineering. It relies on extensive textual description and the use of the wide range of diagramming methods offered in the Unified Modelling Language (UML). These documents and diagrams record the understanding and imaginative work of participants in the business and of system analysts. When bringing EM methods to bear upon problems of BPR an important difference in comparison with a UML approach is that these early conceptual visions and insights can be directly supported and embodied in the building of our computer artefact. Documents and diagrams may, of course, still be important for contractual, auditing and explanatory reasons and these can be developed in parallel with the artefact construction. But the benefits of having an evolving and shareable computer model from the outset of conceiving a system – for the sake of commmunication, for monitoring and validating requirements, for smooth system development, and so on – are obvious and substantial.

The use case model is a major part of the requirements model in OOSE. The elaboration and maintenance of the requirements for a system is central for any further development. It plays a key role at both the cognitive and operational levels. We have always seen the issue and challenges of requirements for modern

systems as an area of natural application for EM to conventional software. We now turn in the next section to an EM perspective on the evolution of requirements.

4.2 The SPORE Framework

A major theme of this paper is the need to consider the larger context of the processes we are interested in and are modelling. In a business application, this means including the objectives of an organisation, the viewpoints of the people concerned in any particular process, and the motives, knowledge and expectations of users of systems being considered. The application of EM requires us to widen our focus from an intended computer system to include the entire business processes (the environment) and the people involved (the human factors). It also means shifting our first attention from software requirements to business requirements. The former should be a result of the latter.

The framework that is described in more detail in this section is primarily directed towards requirements. But by 'requirements' we do not mean the elicitation and formulation of required behaviour. Instead we have in mind the *embodiment* of the requirement. That is, a computer model which exhibits, through visualisation and interaction, the behaviour and features of the system or solution required. We are not primarily concerned here with textual specification of the requirement (although this would not be difficult to produce on the basis of such a model). It is a feature of EM that the conventional phases of system development (specification, design, implementation, testing etc) tend to be conflated and are continuously elaborated during the evolution of our models. It is for this reason that we sometimes speak of 'cultivating requirements' in the same breath as 'building an ISM'; they are the same process viewed from different perspectives. So the building of an ISM is somewhat like building a prototype – although not one that is thrown away – but one that is elaborated and can be refined and optimised into a final useful system.

SPORE is a problem-oriented framework in which requirements – viewed as solutions to the problems identified in the application domain – are developed in an open-ended and situated manner. Within this framework, people participating in the requirements engineering process are able to cultivate requirements through collaborative interaction with each other aimed at solving the identified problems, rather than searching for requirements from the 'jungle' of user's needs (Sun et al., 1999). For a given application, a family of artefacts or interactive situation models (ISMs) are developed which form the medium for the problem-solving process of requirements cultivation.

The SPORE framework for building situated models for the requirements engineering process is depicted in Figure 3. The inputs of the SPORE model are:

- *Central problems* of the domain which are identified by the participants with reference to their concern for the functional and non-functional requirements of the developing system. The identification of problems can occur at any time during the process and is rarely regarded as completed.
- *Relevant contexts*, such as the organisation's goals and policy and the relationships between participants, act as motives and constraints for the participants in creating the outputs.
- *Available resources*, such as documents, technology and past experiences of participants, are used to facilitate the creation of the SPORE model's outputs.

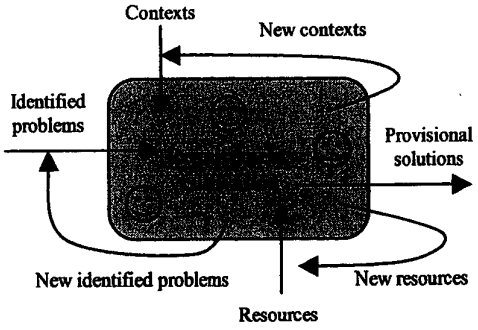

Figure 3. The SPORE framework

The four kinds of outputs from the SPORE model are: *provisional solutions* which are developed by participants on the basis of the available resources and the relevant contexts. The other outputs, including *new contexts*, *new resources* and *new problems*, combine with their earlier versions and form new inputs for creating the next output. That is, all these contexts, resources and problems, even during the development of solutions, always remain modifiable and extensible. In view of this, participants can develop requirements in a situated manner to respond to the change in the contexts, resources and even the problems themselves. Thus this framework addresses the fact that requirements may be changing all the time and can rarely be regarded as complete.

The nature of the SPORE framework is iterative and incremental which means that the ISMs are built in a sequence of structured development cycles, each of which is adding a new portion to the whole model. The delivery of small increments allows continuous feedback and evaluation of the progress achieved.

This experimental interaction is particularly powerful because the participants can interact with each other as well as with the model. Using network facilities the interaction of a participant can be propagated to the artefacts of other participants and consequently affect their insights. Within SPORE all computer models of participants can be connected together. When definitions are propagated they first change the visualisation of other participants' artefacts (given suitable authorisation) and subsequently may alter their insights as well. So participants can collaboratively interact with each other through their artefacts.

In such a collaborative environment, a working understanding of the key problems and their solutions,

i.e. requirements, can be established. This working understanding can then be cultivated, i.e. grown incrementally, through the successive interaction between participants for exploring and integrating individual insights. On the whole, greater consistency between the individual insights indicates improved mutual understanding. For this reason, participants will continually refine their interaction with a view to achieving greater coherence and consistency.

4.3 Using SPORE for Participative BPR

We have emphasised in the previous section the importance of gaining a shared understanding of problems – or at least a working understanding of them – in order to negotiate towards consensus and agreement on their solutions. No doubt people and organisations have always sought to do this, but predominantly by using natural language, diagrams and, possibly, physical artefacts. We suggest that computers, when viewed in the broad perspective described in section 3, are now allowing us a new means for sharing understanding and knowledge. What we are aspiring to do in EM, and with the SPORE framework, is to introduce a powerful electronic modelling medium for the shared construction of artefacts that can faithfully and flexibly embody existing, and planned, real world systems. This embodiment, which exploits the physical aspects of computers, has far-reaching consequences. There is a continuous evolution possible from an original conception of a system to a useful developed form. It also means this evolution is at all times open to revision and interaction, with immediate feedback. The feedback may be in the form of direct experiential knowledge as well as more conventional propositional and mathematical representations. Any EM model with visualisation is such an embodied artefact, or an ISM as we have described it in earlier sections (3.2 and 4.2).

A family of ISMs built in the SPORE framework can be regarded, according to the patterns of interaction invoked, as a *requirement* (when we interact in the roles of particular users), or a *system* (when we interact in the 'roles' of key components, or agents in the EM sense, and so are exploring the internal structure of the model), or a *business process* (when we interact in the roles of workers, markets, suppliers etc). It is such flexibility of interpretation – according to style of interaction – that allows the SPORE framework to be naturally applied to BPR.

At the end of each subsection of section 2, we gave some hints as to how EM might contribute to BPR. We now gather those together in the light of the above. For 2.1 we have already indicated the unusual flexibility of ISMs to accommodate unforeseen and arbitrary changes in an environment. In 2.2 we stressed the importance of the key problem that many have diagnosed with conventional BPR: namely the difficulties, and yet necessity, of truly involving all relevant people in a reengineering process. EM is a human-centred approach which now has tools supporting distributed working with sophisticated modes of communication. The same benefits of working participatively on requirements that we mentioned in 4.2 naturally carry over to any BPR application. Many of the features of business process modelling that are highlighted in 2.3, for example, making dynamic behaviour explicit, being able to communicate it and analyse it, have been addressed already, albeit briefly. The use of modelling for simulation and control of automated workflow can be seen in our case study example in section 5.3. The framework of problem situation and solution space also described in 2.3 as appropriate to BPR is adopted in SPORE. Finally, the need to implement reengineered processes by means of building systems, and the way this has been done using OO methods are sketched in 2.4. The reservations expressed there about OO methods and the need for a wider context have now been amplified. We believe we can derive useful systems directly from our artefacts although we have only limited experience so far of doing so, and only with small scale examples. Some indications of how this can be achieved are given in (Beynon et al., 2000b).

We now proceed to a more detailed consideration of a practical example of using EM methods for the analysis of a problem and construction of an ISM. This could be the basis of establishing part of a requirement for a system, for the reengineering of existing processes in a business, and for the development of an associated useful system.

5 EM FOR A WAREHOUSE SYSTEM: A CASE STUDY

EM offers an open-ended environment that provides an alternative approach both to business modelling and to system development. It also promotes the participation of users (stakeholders) in the business modelling activity. Since the stakeholders share a common interest in the success of the business, it is important that our framework supports sharing and distribution of information and knowledge, as well as the learning and experimentation that contribute to the continuous evolution in its organisation.

A warehouse management system is taken as a case study to illustrate the potential for applying the SPORE framework in BPR. This case study was adopted by Jacobson et al. in their text *Object-Oriented Software Engineering* (1992). At that time, their main concern was to identify the requirements of proposed computer systems by use case analysis and modelling. The idea behind the use case approach is that if we understand the roles of the users who need access to the system, then we shall identify some of the essentials from which requirements are elicited. For Jacobson, each use case is associated with a particular kind of interaction between human agents (actors) and the system, such as might be directed towards one of the required functions of the warehouse (e.g. manual redistribution between warehouses). In a subsequent book, *The Object Advantage*, Jacobson et al. (1995) extend the use case approach to modelling business processes by introducing the concept of a 'business use case', and propose a method for object-oriented business engineering. There the use case model serves as a process model of the

existing business (the outside view of the company), which is used as the basis for prioritising the processes to be reengineered.

In this paper, we also regard it as important to address BPR in the broader context of developing a business process model. This means widening our focus to include the 'real world' (the environment and human factors) rather than the computer system alone. When we adopt this perspective, the role of EM is to develop a computer-based model that can be used to explore all the characteristic transactions of the warehouse. To this end, the character of our framework is through-and-through agent-oriented, so that the warehouse activity is conceived with reference to state-changing protocols for human and automated components with the system. In as much as human actions are constrained by the business process and follow some reliable patterns, it is possible to regard their co-operative activity as a form of computation (in the same way that we might say "the users are programmed"). The human users and computer-based components can then be viewed as computational agents in a complex system. This agency is mediated through the user-computer interface: the input of the user influences the state of the computer, and the output of the computer changes the environment of the user (Beynon and Russ, 1994).

5.1 Introduction to the Warehouse Example

This case study involves applying the SPORE framework to a warehouse management system to achieve BPR. Our goal is to illustrate the use of the EM concepts discussed earlier, but it is not possible to give a complete overview or fully illustrate the entire study.

The proposed system is to support warehouse management. The main function of a warehouse is to provide its customers with warehouse space. The operations of the warehouse also include storing different kinds of items and using trucks to redistribute the items. The aim of introducing computer systems into the warehouse is to offer automatic support to the storage and redistribution services. This involves keeping track of the locations and status of items, differentiating between kinds of items (those that are perishable or flammable), maintaining security and integrity checks, and managing storage, retrieval and relocation. (The possibility that some of the functions of the warehouse associated with the physical storage and retrieval of items might also be automated using robots is not beyond the scope of our approach, but this is not directly addressed here.) The people in the warehouse who will use this system may include: the *foreman* responsible for the warehouse; the *warehouse worker* who is responsible for loading and unloading; the *forklift operator* who drives a forklift in the warehouse; the *truck driver* who drives a truck between different warehouses; the *office personnel* who receive orders and requests from customers, arrange the truck routine, and keep records of all warehouses.

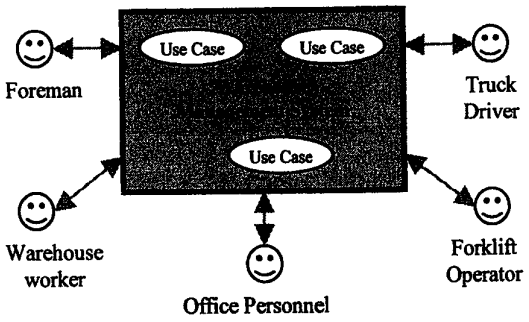

Figure 4. Diagram of the Initial Warehouse System with Actors Identified (source: amended from Jacobson et al., 1992)

5.2 Business Process Model for Warehouse

In applying the traditional use case approach, the first step is to create a simple picture of a system that describes the system boundaries and the actors (users) of the system (cf. Figure 4). The EM approach differs in a significant respect – the boundary of the system is not preconceived but grows with the understanding of the modeller. In conventional system development, because the boundary is defined in advance, the modeller focuses on those interactions that respect the functionality that is imposed on the system. In an EM approach, the warehouse operation is conceived in terms of each agent's perception of states and state changes. For this reason, our initial concern in developing the business process model is with studying the capabilities of the agents that are intended to operate, and examining the possibilities for their unconstrained interaction in an experimental manner. This is the basis for subsequently exploring the protocols these agents can realistically follow in order to carry out the preconceived characteristic transactions of the warehouse.

In traditional modelling approaches for business, the issue of how agents apprehend the current state of a business process is not explicitly addressed. Of course, there are some key observables that are recognised from the outset to be strongly related both to the daily work of personnel within the warehouse and to the phases in preconceived transactions. For instance, in the warehouse case study, these include the items stored in the warehouse and their locations. The EM approach pays much more serious attention to the true character of real-world observation. As a result, the role of observables in the business process differs in three important respects. Firstly, the precise way in which states and events are observed by an agent is considered to be crucially important. It is not simply the fact that an item is at a location, or that a truck has arrived at the warehouse that is deemed significant; it also matters how and by whom the presence of an item or the arrival of a truck is or can be observed. Secondly, the fact that agents are aware of the abstract stage that has been reached in the business process is taken into account in identifying their observables. For instance, the office personnel will distinguish the abstract status of a redistribution process

according to whether a group of items is still at the warehouse, in transit or has now been successfully relocated. Thirdly, there has to be a means by which agents interpret physical observation of real world state as disclosing the status of abstract business transactions. For instance, there must be some concrete indication that is now timely to register that an item has now officially left the warehouse, and that a new phase in the redistribution process has begun.

Our EM business model is framed with reference to the state change of an abstract nature that is associated with observation of a process. For this purpose, the relevant observables relate to the current status in communicating information about characteristic warehouse operations between warehouse personnel. The corresponding state changes are concerned with the systematic execution of protocols and the associated transition from one phase to the next. By using such state changes as a representation for business processes, we will easily identify the existing processes, and (from potential problems or dissatisfaction from customers or employees) can also find those that are candidates for BPR activity. An important aspect of the observables in our business model is that they should not only serve to determine the current state, but must also supply a transaction history appropriate for auditing.

In order to understand the existing business processes properly, the ISM we develop to represent the business process model is modelled on the practices that would have been used in the operation of the warehouse prior to the introduction of computers. (This is consistent with Jacobson's emphasis on the benefits of modelling existing practice (Jacobson et al., 1995).) In this context, forms and paper delivery serve as records of the operation of the model. This kind of manual data entry following systematic processes of form delivery can represent both the current status of all transactions (such as which items were in transit) and the history of transactions. The objective of BPR is to automate these transactions by introducing computer systems, and to try to find alternatives that will, for example, reduce the work-hours of personnel and achieve a more efficient process for business.

From our perspective, the forms can be interpreted as a *paper-based* ISM for the business process. In carrying out a particular transaction, specified procedures are to be followed in filling forms and transferring them between personnel. For instance, as depicted in Figure 5, when a manual redistribution between warehouses is initiated, four copies of redistribution forms (RFs) are transferred from the foreman to the warehouse worker. The manual activities of processing forms effectively identify which agents have roles in the transaction, which are currently active in any phase, and how their interaction is synchronised (cf. Figures 5 and 6). The current status of any transaction is determined by what sections of forms are currently completed and who currently holds the forms.

The modifications that agents make to forms, and the movement of the forms themselves, can be construed as tracing a path through the business process. To elaborate this in more detail we must refer to the observational and interactional context for each agent: the observables it can refer to (its *oracles*), those it can conditionally change (its *handles*) and the protocol that connects these. Note that the relevant observables in this context may refer to the state of the warehouse itself (e.g. an item can be signed off only if it is presently to hand), and relate to the high-level context for interpretation (e.g. issues of legality, safety etc.). The persistence of the record that the forms supply is also significant for auditing and traceability.

Our account of the observational and interactional context makes use of the agent-oriented modelling notation LSD, as illustrated in Listing 1. In this account, the interpretation of agent actions may vary according to the current status of the business process being investigated, and the modeller's current understanding of it. For instance, Beynon (1997) and Ness (1997) identify three views of agents, each appropriate to a different context. In the early stages of familiarisation with an environment or putative system, an agent has unexplored potential to affect system[1] state (view 1). At a later stage, an agent may be construed as reliably following some particular patterns of stimulus-response within the system (view 2). When an appropriate business process has been successfully identified and implemented, each agent enacts a pattern of stimulus-response interaction that can be entirely circumscribed and predicted (view 3). In view 1 our concern is whether an entity has any influence over its environment and in view 3 our concern is whether the exact nature of the influence is known. The term "agent-oriented" is commonly used to refer to activities that are being interpreted from the view 3 perspective (Shoham, 1993), but EM promotes the idea that the concept of agency is only meaningful in relation to the development of understanding from view 1 to view 3 perspectives.

In our warehouse case study, there are contexts in which all the human agents can be viewed in each of these various ways. Whilst the modeller is initially unfamiliar with the environment and the processes of the business, the personnel will represent examples of view 1 agents whose interaction with the warehouse environment and the operation of its business is as yet unexplored. When the roles of a particular employee, such as foreman, have been more clearly identified, they can be regarded as view 2 agents whose pattern of stimulus-response can be in some respects clearly identified. The aim of our modelling process is to fully understand the whole business process and attribute automatic agency to view 3 agents whose pattern of stimulus-response is entirely predictable. This accords with our thesis that the business model is a form of generalized program, and that business process reengineering closely resembles program requirements capture.

---

[1] The 'system' is our warehouse case study is regarded as the whole organisation, not only the computer system itself.

The LSD account can be viewed as identifying and classifying the observables that capture the modeller's current understanding of the warehouse operation. If experimental interaction with the model in due course justifies the transition from a view 1 to a view 3 perspective, the LSD account can be regarded as specifying the observables that describe the stimulus-response patterns in the organisation. In an LSD account, observables that are attached to an agent are referred to as *states*. In general, these observables can be directly manipulated by another agent. For example, the agent `warehouseWorker` can change the status of an item to 'moving pending' by manipulating the `rf_moving_pending` observable which is a state for the agent `rf`. The *oracles* are the observables to which an agent responds. For example, `rf_item` and `rf_quantity` in the agent `warehouseWorker` are examples of oracles for the warehouse worker, who has to know the identity and quantity of items to be redistributed before changing their status (cf. Figure 6 and Listing 1). The *handles* for an agent are those observables that are conditionally under its control. The observable `rf_moving_pending` is an example of a *handle* for the agent `warehouseWorker`.

The stimulus-response patterns for an LSD agent are modelled in two ways. The *derivates* are used to represent stimulus-response relationships that are indivisibly coupled. For example the observable `transportationError`, which indicates whether there is truck available for the specific time at which the foreman intends to make redistribution, is a *state* for the agent `environment` but also a *derivate* for the agent `foreman`. That is to say, any change in the status of truck availability will be deemed to simultaneously change this observable. Looser coupling of stimulus and response is modelled in *protocols*, which consist of a set of guarded actions, each of which takes the form of an enabling condition and an associated sequence of redefinitions of observables. Each guarded action can be regarded as a privilege to act. That is, if an enabling condition pertains, a particular action may be performed. As an example of this principle, the agent `warehouseWorker` receives redistribution forms from the foreman (the enabling condition), then decides the loading time and platform, and passes the forms to both the office and forklift operator (the guarded action).

5.3 The ISMs for the Warehouse State

In interpreting the business process model, and ensuring that its abstract phases are or can be appropriately embodied in agent perception and action, it is essential to take account of the physically explicit observables associated with the warehouse. In keeping with the situated nature of SPORE framework, the ISM is used to incorporate the matter-of-fact observations of the current state of the warehouse. As mentioned earlier, typical observables that are significant in this view are the items and locations in the warehouse, and the inventory that connects items with locations. An ISM to represent these observables will supply visual representations for items and locations, and display the status of the inventory. Such a representation of the current warehouse state is complemented by informal actions, such as represent the relocation of items, item look-up in the inventory or receipt of a new item for storage. In some contexts, this will motivate visualisations to represent intermediate states in the operation of the warehouse, associated with items in transit, or items located via the inventory but yet to be retrieved from the warehouse.

A model of the warehouse has to incorporate such aspects of state and state change in order to be faithful to its referent. It must also provide the setting in which to consider behaviours that are undesirable or outside the scope of normal operation. Relevant observables required for this purpose might address issues such as the loss of items or warehouse locations, the concept of items being mislaid, or of items being perishable.

There is no single ISM that can represent all the aspects of the warehouse state. The state of the warehouse will typically be represented by different ISMs according to what problems are being addressed in the SPORE framework. The scale of an ISM is limited by the number of definitions that can be conveniently stored, rapidly accessed and efficiently processed, but in these respects it is well-suited to those concerns that lie within the modeller's conceptual grasp. The ISM we construct here incorporates the 'seed' ISMs for the warehouse: the form-based abstractions that capture the state of the business process model and the activities of the agents; the storage, retrieval and distribution of items; and additional observations such as are associated with the wider significance of the warehouse operation (e.g. concerned with the legality and the integrity of the business process). The potential framework for BPR established by applying SPORE is illustrated by the transformation from a paper-based to a computer-based ISM.

The distributed version of EDEN enables us to separate the viewpoints of the agents in the model, and to complement these with an external observer's interpretation. Figure 6 illustrates how computer-based forms are used to represent the environment for each agent's interaction. In this way, the distributed ISM can serve as a medium in which to identify and enact appropriate transactions, and to debug and refine these through collaborative interaction between the various participants. Many possible issues in requirements can be addressed by SPORE in this way:

- Through experimentation at different workstations, we can identify issues that are problematic from the perspective of particular agents: for instance, "how does the office know which drivers are available?", "how does the office determine whether a transaction is completed?".
- Through the elaboration of different seed ISMs, we can address additional issues, such as transportation costs, perishable goods, security and trust concerns.
- Through modifying dependencies and communication strategies, we can consider the effects of different technologies, such as are associated with the use of mobile communications,

the Internet, optical bar code readers, or electronic locking agents.

- Through collaboration and synthesis of views, we can distinguish between subjective and objective perceptions of state e.g. to contrast "I remember doing X" with "I have some record of doing X" with "There is an official record of X", or to model misconceptions on the part of an agent.
- Through intervention in the role of superagent, it is possible to examine the consequences of singular conditions that arise from opportunistic interaction or Acts-of-God, and to assess activities outside the scope of normal operation such as are associated with fraud, or manual back-up to automated procedures.

6 Conclusions

We have introduced a novel approach to modelling that is based on a view of computation and programming that is significantly broader than conventional views. On this view (EM), based as it is on the concepts of observable, dependency and agency, computer-based models of business processes can be built in a way similar to that in which humans make conceptual models of such processes. We can then specialise and circumscribe our models to derive traditional software systems. In this way EM can offer both cognitive and operational support to BPR from the very early, conceptual stages of modelling. The potential benefits of introducing the SPORE framework for reengineering a business process are its flexibility, openness and the richness of interaction possible between many human participants in the modelling environment. Our case study shows the potential for modelling current practice in a business. There is clearly much future work to be done on exploring the scalability of our approach and the derivation of practical systems from our models.

7 ACKNOWLEDGEMENTS

The authors would like to acknowledge colleagues of the Empirical Modelling Group for the many constructive suggestions they have made, and Pi-Hwa Sun for his work on a distributed version of EDEN and the initial idea of the SPORE framework. The third author is indebted to the EPSRC for funding via the AMORE project, grant M02675.

8 REFERENCES

Beynon, W.M. (1997) "Empirical Modelling for Educational Technology," in *Proceedings of the Second International Conference on Cognitive Technology*, IEEE, August 1997, University of Aizu, Japan, pp.54-68

Beynon, W.M., Rasmequan, S., Russ, S.B. (2000a) "A New Paradigm for Computer-Based Decision Support," submitted to *Journal of Decision Support Systems*

Beynon, W.M., Rungrattanaubol, J., Sinclair, J. (2000b) "Formal Specification from an Observation-Oriented Perspective," to appear in *J-UCS*

Beynon, W.M., Russ, S.B. (1994) "Empirical Modelling of Requirements," Research Report 277, Department of Computer Science, University of Warwick

Davenport, T.H. (1993) Process Innovation: Re-engineering Work through Information Technology, Boston: Harvard Business School Press

Davenport, T.H. (1996) "Why Re-engineering Failed: The Fad that Forgot People," in *Fast Company*, Premier Issue, pp.70-74

Ellis, T.I., Dooner, M., Swift, K.G. (1997) "Better the Devil You Know – the Role of Pragmatic Modelling," in *First International Conference: Managing Enterprises – Stakeholders, Engineering, Logistics, and Achievement (ME-SELA'97)*, 22-24 July 1997, Loughborough University, UK, pp.151-156

Galliers, B. (1998) "Reflections on BPR, IT and Organisational Change," in Galliers, R.D., Baets, W.R.J. (eds.) Information Technology and Organisational Transformation, Chichester, England: John Wiley and Sons Ltd, pp.225-243

Gerrits, H. (1994) "Business Modelling based on Logistics to Support Business Process Re-engineering," in (Glasson et al., 1994), pp.279-288

Glasson, B.C., Hawryszkiewycz, I.T., Underwood, B.A., Weber, R.A. (eds.) (1994) Business Process Re-engineering: Information Systems Opportunities and Challenges, Amsterdam: Elsevier Science

Hammer, M., Champy, J. (1993) Reengineering the Corporation: A Manifesto for Business Revolution, New York: Harper Business

Hutchison, A. (1994) "CSCW as Opportunity for Business Process Re-engineering," in (Glasson et al., 1994), pp.309-318

Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G. (1992) Object-Oriented Software Engineering: A Use Case Driven Approach, Harlow, England: Addison-Wesley

Jacobson, I., Ericsson, M., Jacobson, A. (1995) The Object Advantage: Business Process Reengineering with Object Technology, Wokingham, England: Addison-Wesley

Jacobson, I., Griss, M., Jonsson, P. (1997) Software Reuse – Architecture, Process and Organisation for Business Success, Harlow, England: Addison-Wesley

Kaindl, H. (1999) "Difficulties in the Transition from OO Analysis to Design," in *IEEE Software*, September/October 1999, pp.94-102

Ness, P.E. (1997) Creative Software Development: An Empirical Modelling Framework, PhD Thesis, Department of Computer Science, University of Warwick, October 1997

Nurcan, S., Grosz, G., Souveyet, C. (1998) "Describing Business Processes with a Guided Use Case Approach," in *Proceedings of the 10th International Conference (CaiSE'98)*, June 1998, Pisa, Italy.

Rasmequan, S., Roe, C., Russ, S.B. (2000) "Strategic Decision Support Systems: An Experience-Based

Approach," in Proceedings of the 18th IASTED Conference on Applied Informatics, February 2000, Innsbruck, Austria

Riemer, K. (1998) "A Process-Driven, Event-Based Business Object Model," in *Proceedings of Second International Workshop in Enterprise Distributed Object Computing (EDOC'98)*, 3-5 November 1998, La Jolla, California, USA, pp.68-74

Schader, M., Korthaus, A. (1998) "Modelling Business Processes as Part of the BOOSTER Approach to Business Object-Oriented System Development Based on UML," in *Proceedings of the Second International Enterprise Distributed Object Computing (EDOC '98) Workshop*, 3-5 November 1998, La Jolla, California, USA, pp.56-67

Sherwood-Smith, M. (1994) "People Centred Process Re-engineering: An Evaluation Perspective to Office System Re-design", in (Glasson et al., 1994), pp.535-544

Shoham, Y. (1993) "Agent-Oriented Programming," in *Artificial Intelligence*, Vol. 60, No. 1, pp.51-92

Simons, A.J.H. (1999) "Use Cases Considered Harmful," in Mitchell, R., Wills, A.C., Bosch, J., Meyer, B. (eds.) *Proceedings of the 29th Conference Tech. Object-Oriented Programming Language and Systems (TOOLS-29 Europe)*, Los Alamitos: IEEE Computer Society, pp.194-203

Simons, A.J.H., Graham, I. (1998) "37 Things that don't Work in Object Modelling with UML," in Kent, S., Mitchell, R. (eds.) *British Computer Society Object-Oriented Programming Systems Newsletter*, Vol. 35, Autumn 1998

Simons, A.J.H., Graham, I. (1999), "30 Things that Go Wrong in Object Modelling with UML 1.3," in Kilov, H., Rumpe, B., Simmonds, I. (eds.) Behavioral Specifications of Businesses and Systems, Kluwer Academic Publishers, Chapter 16, pp.221-242

Sun, P.H. (1999) Distributed Empirical Modelling and its Application to Software System Development, PhD Thesis, Department of Computer Science, University of Warwick, July 1999 (also can be downloaded from http://www.dcs.warwick.ac.uk/modelling)

Sun, P.H., Chen, Y.C., Russ, S.B., Beynon, W.M (1999) "Cultivating Requirements in a Situated Requirements Engineering Process," Research Report 357, Department of Computer Science, University of Warwick

van Meel, J.W., Bots, P.W.G., Sol, H.G. (1994) "Towards a Research Framework for Business Engineering," in (Glasson et al., 1994), pp.581-592

Vidgen, R., Rose, J., Wood, B., Wood-Harper, T. (1994) "Business Process Reengineering: The Need for a Methodology to Revision the Organisation," in (Glasson et al., 1994), pp.603-612

Vogel, D. (1994) "Re-engineering Towards the Meeting of the Future", in (Glasson et al., 1994), pp.35-45

Warboys, B., Kawalek, P., Robertson, I., Greenwood, M. (1999) Business Information Systems: A Process Approach, Berkshire, England: McGraw-Hill Publishing

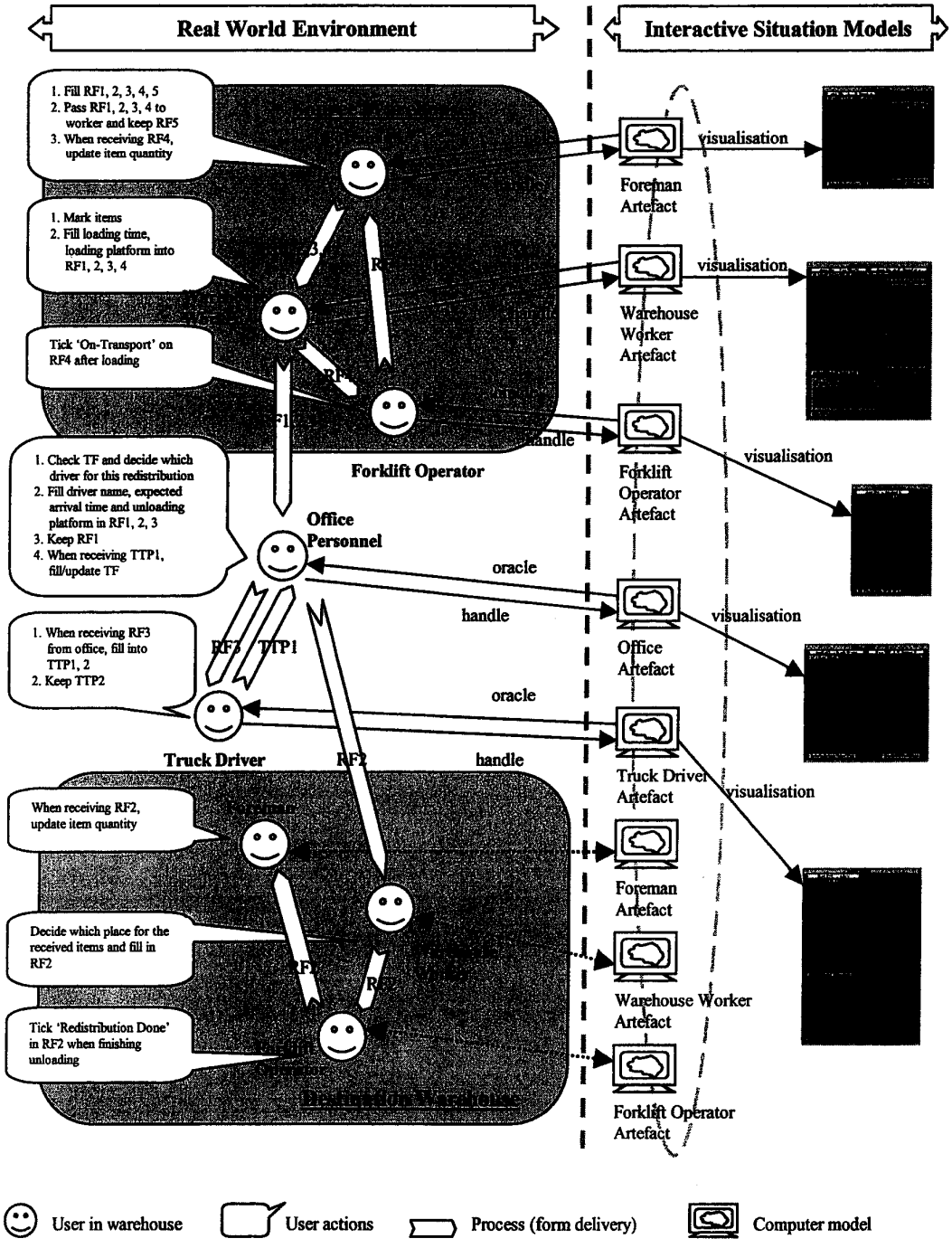

Figure 5. A collaborative working environment for manual redistribution between warehouses

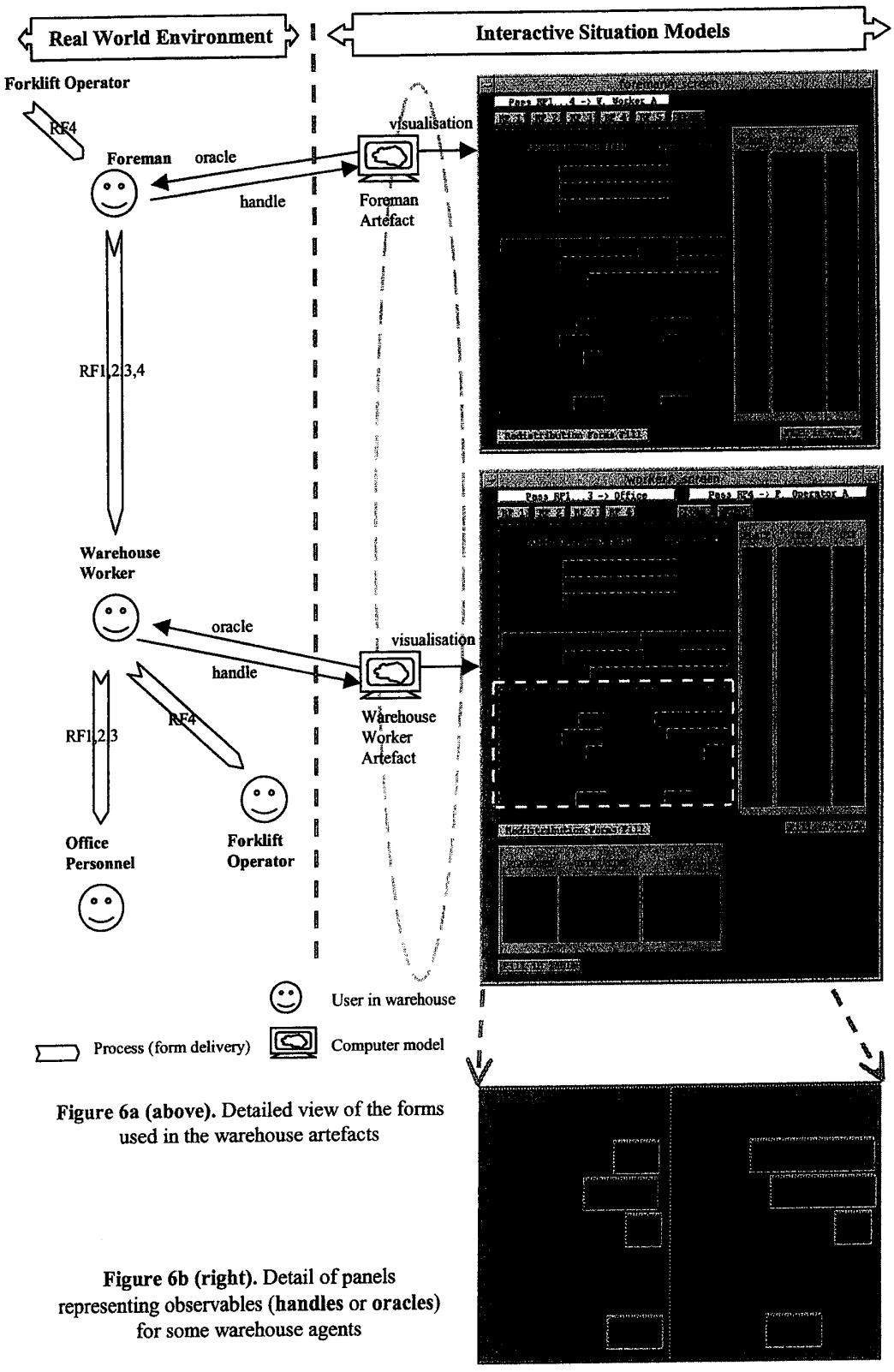

Figure 6a (above). Detailed view of the forms used in the warehouse artefacts

Figure 6b (right). Detail of panels representing observables (**handles** or **oracles**) for some warehouse agents

Listing 1. Part of the Outline LSD Specification for the Warehouse Management System

```
AGENT warehouseWorker(w) {
   STATE
   ORACLE
      rf_item[1...4],
      rf_quantity[1...4],
      rf_from_place[1...4],
      rf_date[1...4],
      rf_item[2],
      rf_quantity[2],
      rf_date[2]
   HANDLE
      rf_moving_pending[1...4],
      rf_loading_time[1...4],
      rf_loading_platform[1...4],
      rf_redistribution_confirm[1...4],
      rf_to_place[2],
      rf_redistribution_confirm[2]
   DERIVATE
   PROTOCOL
      *** Receiving RF1...4 from foreman ***
            --> read rf_item[1...4]; read rf_quantity[1...4];
                read rf_from_place[1...4]; read rf_date[1...4];
                write rf_moving_pending[1...4];
            --> write rf_loading_time[1...4];
                write rf_loading_platform[1...4];
            --> *** Pass RF1,2,3 to office and RF4 to forklift operator; ***
                                                  // For loading
      *** Item not enough ***
            --> write rf_redistribution_confirm[1...4]
                   ==('Error'&&'Item not enough');
            --> *** Pass RF1...4 back to foreman; ***
      *** Receiving RF2 from office ***
            --> read rf_item[2]; read rf_quantity[2]; read rf_date[2];
            --> write rf_to_place[2];
            --> Pass RF2 to forklift operator; ***
                                                  // For unloading
      *** All places full ***
            --> write rf_redistribution_confirm[2]
                   ==('Error'&&'All places full');
            --> *** Pass RF2 back to foreman; ***
}

AGENT rf(n=1...5) {                               // Redistribution Form
   STATE
      rf_warehouse[n]=@,                          // Information in RF1,2,3,4,5
      tf_foreman_name[n]=@,                       // Information in RF1,2,3,4,5
      rf_job_number[n]=@,                         // Information in RF1,2,3,4,5
      rf_item[n]=@,                               // Information in RF1,2,3,4,5
      rf_quantity[n]=@,                           // Information in RF1,2,3,4,5
      rf_from_place[n]=@,                         // Information in RF1,2,3,4,5
      rf_to_warehouse[n]=@,                       // Information in RF1,2,3,4,5
      rf_date[n]=@,                               // Information in RF1,2,3,4,5
      rf_moving_pending[n]=@,                     // Information in RF1,2,3,4
      rf_loading_time[n]=@,                       // Information in RF1,2,3,4
      rf_loading_platform[n]=@,                   // Information in RF1,2,3,4
      rf_driver[n]=@,                             // Information in RF1,2,3
      rf_arrival_time[n]=@,                       // Information in RF1,2,3
      rf_unloading_platform[n]=@,                 // Information in RF1,2,3
      rf_on_transport[n]=@,                       // Information in RF4
      rf_to_place[n]=@,                           // Information in RF2
      rf_redistribution_confirm[n]=@              // May appear in RF1,2,3,4
   ORACLE
   HANDLE
   DERIVATE
   PROTOCOL
  }
```