Cultivating Requirements in a Situated Requirements Engineering Process

P.H. Sun, S.B. Russ, Y.C. Chen and W.M. Beynon

Department of Computer Science
University of Warwick
Coventry CV4 7AL, UK

{sun, sbr, cheny, wmb}@dcs.warwick.ac.uk

ABSTRACT

The fact that requirements are situated motivates an alternative to conventional process models that gives adequate recognition to the situatedness of the requirements engineering process (REP). This paper proposes a problem-oriented framework SPORE (situated process of requirements engineering) whereby requirements as solutions to the identified problems in the application domain are developed in an open-ended and situated manner. Within this framework, people participating in the REP are able to cultivate requirements through collaborative interaction with each other for solving the identified problems, rather than to search for requirements from the jungle of users' needs. For a given application, a family of artefacts or interactive situation models (ISM) are developed which form the medium for the problem-solving process of requirements cultivation. These ISMs are built using the principles and tools of the Empirical Modelling, a novel approach to computer-based modelling. Participants can create and use these models not only as artefacts to explore, expand and experience the solutions to the identified problems, but also as a powerful means of supporting their collaborative interaction for 'growing up' the solutions in a distributed environment. A case study of applying this framework to cultivate requirements for a warehouse distribution system is given.

1. Introduction

In spite of their importance, requirements and the processes by which they are apprehended and acquired, are poorly understood [Fin94, Bub95]. Current understandings of requirements engineering are dominated by phase-based models in which a degree of rational planning through a rigid sequence of prescribed phases is assumed [LK95, JP94]. The character of each phase reflects engineering practice, i.e. the application of proven methods, techniques and tools in a systematic and cost-effective fashion. However, such step-by-step algorithms for the requirements engineering process (REP) are of limited use in the rapidly changing domains of modern applications [Gog96, Rya95, SS96, KS98]. Instead it is flexibility and openness to the environment and differing viewpoints that characterise the way in which people engage in situated actions for a common purpose within particular social and organisational contexts [Suc87]. This contrast highlights the importance of recognising the situatedness of the REP in the sense that it can only be fully understood in terms of the particular circumstances which hold in specific cases at specific times.

There is increasing consensus that requirements are not usually pre-existent and hidden in the experts' head waiting to be dug out and put into the specification cabinet [BCDS93]. Neither can they be completely described in any form of logical algorithm. In contrast, requirements are designed and developed through the participants' interaction [Bub95] and are always liable to change. The simple distinction between 'what' and 'how' (traditional in discussing specification and implementation) is inappropriate and inadequate [Dav93, SB82, SS96] because complex requirements are rarely complete and are liable to evolve faster than the REP itself [Bub95].

In addition, most methods for developing the REP aim at achieving an agreed set of requirement specifications in text and diagrams. They seek to represent the informal information elicited for requirements in a detailed documentary fashion. However, paper is passive and can only serve as a repository for the collected information. It is hard for users and developers to know whether or not there are differences between their interpretations of the same text. Many users sign off requirement specifications without fully understanding the implications. It is usually difficult for users to validate the technical documentation used by developers and designers. Users need a system capable of solving their problem in practice, rather than specifications describing the system in an abstract manner. The misunderstanding of requirement specifications can lead to huge additional cost in later stages of system development, or even the failure of the whole system [STM95]. To bridge the communication gap between participants, we are proposing an interactive computer-based environment for building detailed models of a domain (not simply prototypes). Such models are open-ended (the possible interactions are not pre-conceived) and can assist participants in exploring, understanding and creating knowledge in the course of collaborative interaction [Fis91, DS97, Cro94].

Many process models also fail to support group work; the latter is particularly important for the REP, since many stakeholders are involved in the development of requirements. In this connection, it is useful to develop a model for supporting distributed work in the REP not only to allow different participants to work on a common model but to allow multiple viewpoints to be shared and integrated in a well-controlled manner.

The challenge that is addressed in this paper is that of providing an alternative framework for the REP which recognises the situatedness of the process and provides participants with computer support for artefacts. Through these artefacts they can experience the domain model, represent multiple viewpoints, and develop an agreed requirement that is embodied in an artefact that complements the documentary record.

In Section 2, a human-centred, computer-supported framework for conducting the situated process of requirements engineering (SPORE) is proposed. Within this framework, developing requirements is viewed as a problem-solving process in some domain or application. Participants must interact with each other to shape requirements on the basis of their understanding of the domain, their context and the available resources. The need for creating and using computer-based models as artefacts by participants to support their interaction with the domain and with each other is explained in Section 3. These models can be used for exploring and integrating individual insights into the evolving requirements. A case study of using the SPORE framework to develop requirements for a warehouse distribution system is given in Section 4.

2. The SPORE Framework

Since a requirement can be defined as a condition or capability that must be met or possessed by a system to satisfy the condition or capacity needed by a user to *solve a problem* or achieve an objective [IEEE90], it is plausible to view requirements as providing solutions to identified problems. The REP begins in the problem domain associated with the requirements of the developing system. This domain is generally informal, situated and open to the real world [Gog96, Blu93]; it cannot therefore be completely specified in advance. Instead we represent the domain by a situated, provisional, subjective, but computer-based, model. It is *situated* because it is represented as organically connected to its referent (the domain). Such connection is achieved by being continuously open to revision through comparison between the experiences of interaction with the domain and those of interaction with the model. Thus the model is not divorced from the domain as required for a preconceived 'system' with boundaries made sharp by some form of idealisation or abstraction. To extract a useful application we shall eventually have to 'freeze' our model into such a system.

A human-centred framework, called SPORE, for building situated models for the process of requirements engineering is depicted in Figure 1. Key problems of the domain are identified by the participants within the grey box in Figure 1 with reference to their concerns for the functional, non-functional and enterprise attributes of the developing system. The identification of problems can occur at any time during the REP and is never regarded as completed. Another two inputs of the SPORE model are the current contexts and the available resources. The resources, such as documents, technology and past experiences of participants, are used by participants to facilitate the creation of the SPORE model's outputs. The 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. These three kinds of input may impact on different parts of the model at different stages of its evolution. The arrows ending at the inside of the grey box in Figure 1 convey this idea.

There are four kinds of outputs from a SPORE model. The most important one consists of solutions to the identified problems. These are developed by participants on the basis of the available resources and the current contexts. Moreover, 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 to say, all these contexts, resources and the identified problems, even during the development of solutions, are still 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. This implies that requirements are apt to change all the time and thus are never completed. In this respect, the SPORE framework is consistent with J. Goguen's concern for the situatedness of requirements [Gog94].

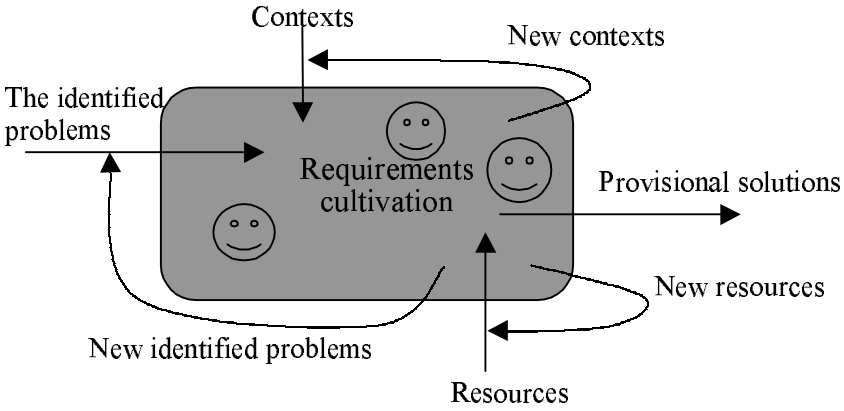

Figure 1. The SPORE framework

The SPORE framework determines neither specific activities nor their sequence. In many cases several problems can be identified simultaneously. Some may be very difficult to solve under the current contexts and resources, but the others are not. Some are interdependent and need to be solved concurrently, but some can be solved independently. Different problems are likely to need solution by different methods. No rule or algorithm can be postulated in advance to take all these factors into account. A generic strategy for taking actions is 'divide and conquer', where the highest priority is to undertake action for the easiest problem. But this is not a golden rule. Participants still must take their current context and available resources into consideration in order to cope with diverse issues raised from the development of solutions.

The central activity in SPORE framework is the requirements cultivation in which participants interact with each other and with their environments to develop requirements, i.e. the emerging solutions to the identified problems. The term 'cultivation' is used to convey the idea that requirements (like plants) should grow gradually rather than be conjectured from their initially fragmentary, chaotic and rapidly changing states. Some models for the REP assume that requirements are pre-existent but hidden in some sources [LK95, SS97], just like grown plants in a huge jungle. The purpose of building these models is to search for (elicit) the right plants (requirements) from the jungle (available sources, such as documentation and the expertise of users). Typically, the jungle is a mixture of numerous kinds of elements that are fluctuating in response to the changes in their environment. It is clear that searching in such a jungle

for one element, which has never been seen before and might keep changing all the time, remains a very difficult challenge [LK95, Bub95].

The concept of requirements cultivation, unlike searching for requirements in a jungle, refers to the growing of requirements for the developing system by participants themselves through their collaborative interaction. The cultivating process focuses on neither the problem domain nor the solution domain but instead on the interaction by which participants seek to solve the identified problems on the basis of their current context and available resources. For example, consider the development of a simplified automated teller machine (ATM) which contains an embedded software system to drive the machine hardware and to communicate with the bank's customer database. In order to acquire requirements of the software system, a problem of accessing service is identified. Then participants relevant to the identified problem, such as customers, bank staffs, machine designers, database managers, security officers, software designer and so on, must work together to solve the problem. The solution is not located in someone's head but socially distributed across all participants. Also, it is formed and shaped through the iterative and creative activities invoked by participants in their interaction with each other. In this sense, requirements are cultivated by participants through various different purposeful activities.

The work of requirements cultivation can be focused further on individual participants. Within the REP, each participant has his/her own individual insight into the identified problems and their solutions. This insight is based on the participant's various contexts and available resources. It is clear that individual insight is often of limited use and inevitably has a bias. For example, in the ATM example mentioned above, for the reason of security, bank staff may demand more rigorous security checking for access to the service provided by an ATM machine. But from the customer's viewpoint, the convenience of use of the services might be the main concern.

3. Applying Empirical Modelling Principles to SPORE

It is important that the SPORE framework involves the use of computer models as artefacts in order to facilitate the exploration and integration of individual insights in an interactive manner. Each artefact is created on the basis of the principles and tools of Empirical Modelling (EM); this is well-established research project developing a novel approach to modelling with computers [Bey99, Rus97]. It allows the user, i.e. the modeller, to use the computer to create an artefact with something of the character of an engineering prototype. Through experiment and observation, the modeller construes an external situation in terms of the primitive concepts of EM: observables, dependencies, agents and agencies, and concurrently constructs a computer model that metaphorically exhibits similar patterns of these concepts. In this way the evolving insight of the modeller is reflected in the coherence between an explanatory model, or *construal* in the modeller's mind, the physical embodiment of this construal in the computer artefact, and a situation in the external environment.

In creating the embodied computer-based construal, the modeller identifies primitive elements in the problem domain corresponding to the fundamental concepts mentioned above, and records them by introducing appropriate definitions, functions and actions into the computer model. A typical step in this process involves the

specification of a user-defined function f and the introduction of a definition of the form t = f(x,y,z) into an existing script of definitions. From the modeller's perspective, this is "recording a dependency between the observables represented by t, x, y and z". From a computational perspective, the abstract semantics of introducing such a definition is typically similar to introducing a new definition into a spreadsheet to which a visualisation of cell values is attached. In particular, the dependencies amongst observables are automatically maintained through a retrospective revision technique. What is more, unlike traditional programming codes, definitions do not have to be entered and organised sequentially. For these reasons, the construction of such computer-based artefacts is a useful vehicle for exploring and developing insights.

EM is a means of constructing knowledge in an experiential rather than a declarative fashion; the modeller's insight is expressed as a coherence between expectations in the mind and the experiments that can be performed on the computer-based artefact and/or in the external environment. The principle resembles 'what if' experiments with a spreadsheet. The modeller introduces new definitions to impose a change of state upon the embodied construal. Almost simultaneously, the new state of this construal is mediated to the user through the visual interface, and evokes a change of state in the mind of the modeller. When this change of state is consistent with the modeller's expectations, it serves to reinforce the modeller's confidence in the way in which a situation has been construed. When the change of state confounds expectations, the modeller must determine whether the situation has been construed in an inappropriate way, or whether a hitherto unsuspected behaviour has been identified. In the latter case, there is a creative element of discovery that is rarely encountered in conventional modelling. This aspect of EM is particularly useful for the exploration of individual insight.

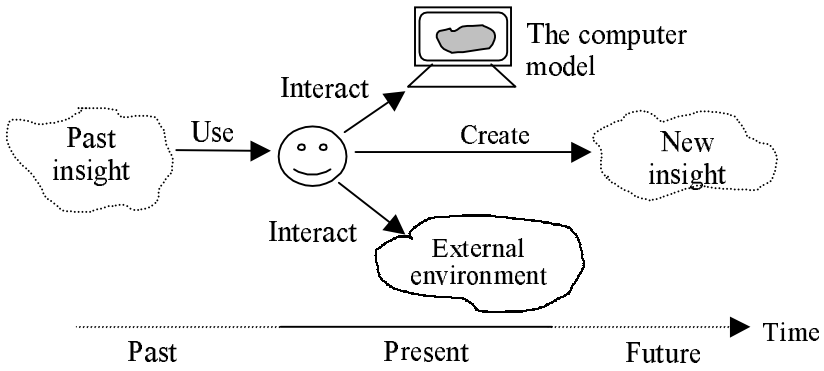

Figure 2. The experimental interaction of a participant

Given the principles and tools of EM, each participant can construct and interact with their artefact to extend their individual insight through 'what if' experiments. The accumulated results of experiments not only change the participant's individual insight immediately but also are stored in the memory of the participant. The latter is the most important resource used by the participant for taking situated actions. In effect, incorporated with other methods, this experimental interaction, using the computer as a modelling medium, can often provide more accurate and powerful resources for

developing solutions to the identified problems. In this sense, interacting with the computer model becomes a very critical situated action for creating a resource for further actions. Figure 2 illustrates that a participant can take situated actions by interacting with their computer model and/or external environment, on the basis of various contexts and available resources, to explore individual insight. The insight into the identified problems and their solutions are evolved with the interaction.

What makes the experimental interaction here particularly powerful is that through it participants can interact with each other. Given the facilities of communication networks, the experimental interaction of a participant can be propagated to others' artefacts and consequently affect their individual insights. Within a networking environment, all computer models of participants can be connected together. Participants can interact with their own artefact privately by making a variety of definitions in order to explore their own insight into the identified problems and corresponding solutions. They also can interact with others and their artefacts by propagating definitions through communication networks. The propagated definitions first change the visualisation of others' artefacts (given suitable authorisation) and consequently may change their insights as well. Thus participants can collaboratively interact with each other through their artefacts and communication networks. Figure 3 shows this collaborative working environment.

Within the collaborative working environment described above, a shared understanding of the identified problems and their corresponding solutions, i.e. requirements, is established. The shared understanding is then cultivated, i.e. grown incrementally, through the successive interaction between participants for exploring and integrating individual insights. Generally speaking, greater consistency between the individual insights is associated with a better shared understanding. For this reason, participants continually refine their interaction with a view to achieving more coherence and consistency. This process is open-ended, and consistency can only be achieved in relation to some restricted work activities and assumptions about reliability and commitment. In practice, there are likely to be singular conditions under which a higher viewpoint must be invoked to mediate or arbitrate where there is conflict or inconsistency. The 'God's view' position shown in Figure 3 indicates such an overall viewpoint. It also could be the view of a requirement engineer when acting in the role of negotiator between differing or incompatible insights.

The most important benefit of interacting with computer models is to make individual insights, and the shared understanding between participants, *visible* and *communicable*. Of course, most models for the REP involve the interaction between participants in order to facilitate the establishment of the shared understanding, for example, by requirements elicitation and validation [LK95]. However, the shared understanding within these models is invisible and incommunicable. Even given a requirements specification, the visibility and communicability of the shared understanding are still restricted to the boundaries of language description and comprehension. Also, paper documentation as used in a repository or archive fails to support the needs of its users in exploring and integrating information. In practice, it is very difficult to keep requirements specifications synchronous to the shared understanding between participants [DS97], because the latter emerges from experimental interaction and evolves much faster than the evolution of specifications.

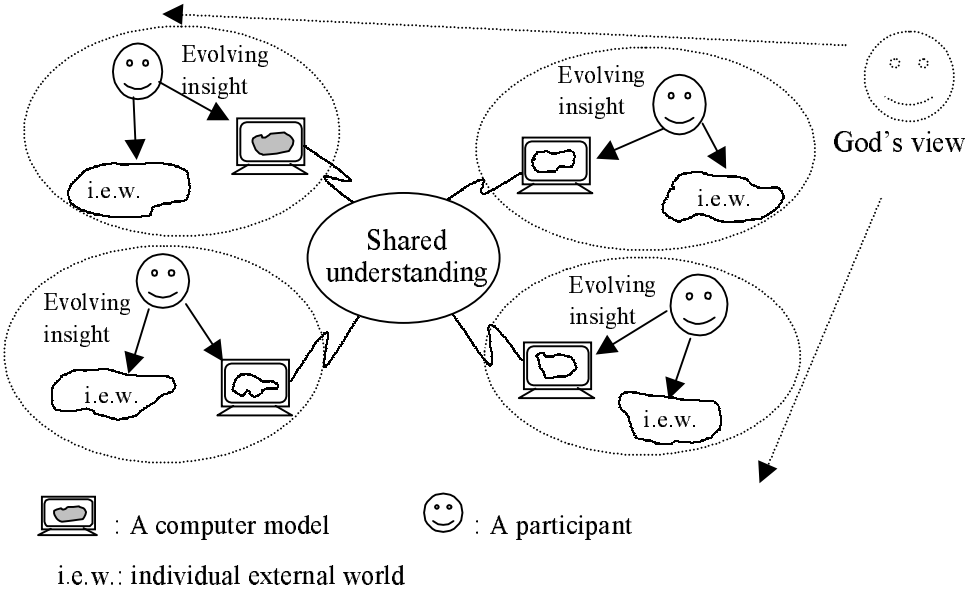

Figure 3. A collaborative working environment for cultivating requirements

In contrast, the experimental interaction between computer models invoked by participants immediately changes the visualisations of these models. The change leads quickly to the evolution of individual insights as well as to a shared understanding. The synchronisation between the evolution of computer models and individual insights allows participants to 'see' the viewpoints of other participants and to 'communicate' with them by interacting with their own artefact. In the same manner, the shared understanding is also shown in these computer models. From the perspective of users, the visible and communicable computer models illustrating the solutions to the identified problems are a crucial contribution to understanding that complements the passive textual descriptions of conventional specifications.

4. Cultivating the Requirements for a Warehouse System

Specifying the requirements for a warehouse is taken as case-study by Jacobson in [JCJO92]. Jacobson's concern is for identifying the software requirements of a computerised system, and his approach is based on use-case analysis. For Jacobson, each use-case is associated with a particular kind of interaction between human agents and the computer system, such as might be directed towards one of the required functions of the warehouse (e.g. manual redistribution between warehouses).

In the context of this paper, the requirements engineering task is seen in the broader context of developing a business process model and determining the role that computer technology can play in the carrying out the characteristic transactions of the warehouse. Our perspective is through-and-through agent-oriented, in the sense that warehouse activity is conceived with reference to state-changing protocols for human and automated components with the system. In effect, where the action of human agents is constrained by the business process so that it follows reliable patterns, it is possible to

regard their co-operative activity as a form of computation. The characteristic transactions of the warehouse are then analogous to use-cases in Jacobson's sense.

4.1 A Tool for Supporting SPORE

In applying SPORE to the solution of this requirements engineering problem, we make use of an interpreter - EDEN - specifically developed to support the principles of EM, as set out in section 3. This interpreter supplies a computer environment that extends and radically generalises a multi-user spreadsheet. It has all the characteristics referred to in the Introduction: supporting group interaction in a distributed environment that is flexible, exploratory in character, and is predominantly based on the use of visual metaphors rather than text. A detailed account of the principles behind EDEN is beyond the scope of this paper, but certain properties are central to the exposition of SPORE that follows.

The computational states in EDEN are suited to the broad view of agency referenced above: state changes can be initiated either by human agents or by automatic procedures. Any particular state, like the current state of a spreadsheet, directly corresponds to a possible state of an external referent. It is specified by a family of definitions (a definitive script) in which the LHS's represent observables, and the RHS's either associate explicit values with these variables, or attach a formula expressing their dependency upon other variables in the script. The values attached to these variables in a script will typically be directly meaningful to the human interpreter because they represent observables in the referent. This correspondence between values of variables and values of observables is generally mediated by visualisation, so that each state of the computer model is directly perceived as metaphorically representing the state of its referent.

An EDEN model of a referent is referred to as an interactive situation model (ISM). ISMs are open to experimental and exploratory interaction resembling interaction with a spreadsheet. There are no sharp boundaries on what interactions are appropriate: an interaction is admissible or of interest provided that it admits some interpretation via its referent. Admissibility is a function of which agent performs the interaction: an ISM commonly admits an agent that can act in a God-like role, such as might be exercised in simulating or resolving unexpected events or anomalous interactions. This role suits the character of participants' interaction within the SPORE framework, which is guided by the specific context and in general follows no preconceived pattern.

As the cultivation of requirements develops, particularly significant patterns of interaction for agents are identified. The roles of agents can be documented as they emerge using notation, which records for each agent the observables that serve as cues for its action (its **oracles**), those which it can conditionally redefine (its **handles**), and what protocol is followed in making this redefinition. A full account of these analyses is beyond the scope of this paper, but the basic concepts are illustrated in Figure 5b, where the panels extracted from forms represent the part played by each warehouse agent in a particular transaction. For instance, the fields specified as "For Worker Use" are handles for the Warehouse Worker, and serve in turn as oracles for the Office Clerk and Forklift Operator.

4.2 Seed ISMs for the Warehouse State

In SPORE, the cultivation of requirements has to start from a representation of those elements of the warehouse state that are pertinent to the particular problem being addressed. This representation will take the form of a *seed* ISM that - because of the situated nature of SPORE - incorporates matter-of-fact observations of the current state of the warehouse. 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 a visual representation for items and locations, and the status of the inventory. (The techniques used to construct such visual representations are not illustrated here, but representative examples can be found on the EM webpage http://www.dcs.warwick.ac.uk/modelling/.)

Such a representation of the current state of the warehouse will be 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. If such aspects are neglected, there is no means to consider behaviours that, though undesirable or outside the scope of normal operation, have a profound influence on the requirement. For instance, the requirements activity has to address matters such as the loss of items or warehouse locations, the concept of items being mislaid, or the significance of perishable items.

There is no single ISM that can represent all the aspects of the warehouse state that are potentially relevant to a requirements identification. The state of the warehouse will typically be represented by different seed ISMs according to what problems are being addressed in the SPORE, and each will be introduced to mimic particular scenarios. For instance, it may be appropriate to construct seed ISMs to represent different varieties of perishable item, or to represent a very large number of items to assess the interface to an inventory database.

4.3 The Warehouse Business Process Model (BPM)

Over and above the naive perception of states and state changes just considered, there is a business perspective upon warehouse operation. This focuses on the particular agents that are intended to operate and the protocols that they follow in carrying out preconceived characteristic transactions. These define the business process model.

The observables in the BPM are different in character from items and locations. They relate to phases in preconceived transactions. The state changes are concerned with systematic execution of protocols and the associated transition from one phase to the next. There may be no counterpart in the BPM for activities that might be possible in practice, such as the illicit retrieval of an item by its owner. An important aspect of the observables associated with the BPM is that they should not only serve to determine the current state, but must also incorporate a transaction history appropriate for auditing.

The ISM we shall develop to represent the BPM is modelled on the practices that were used in the operation of the warehouse prior to the advent of computers. In that context, forms and paper inventories served to record the operation of the BPM, by rendering the abstract observables associated with phases and roles visible and tangible. Manual data entry following systematic processes of form transfer were the means to represent both the current status of all transactions (such as which items were in transit) and the history of transactions.

From our perspective, the forms and inventories can be interpreted as a paper-based ISM for the business process. In performing a particular transaction, specified procedures are to be followed in filling forms and transferring them between personnel. These manual activities effectively identify which agents have roles in the transaction, which are currently active in any phase, and how their interaction is synchronised (cf. Figures 4 and 5). The current status of any transaction is determined by what sections of forms are currently completed and who currently holds the forms.

The full details of how the BPM is construed to operate is reflected in the specific details of what each agent enters on a form. These details refer to the observational and interactional context for each agent: the observables it can refer to (its **oracles**), those is 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.

4.4 Applications of SPORE to Warehouse Requirements

Just as paper records and protocols for interaction with them can be viewed as an ISM, so the process by which such procedures evolved can be construed as EM. The activities involved in this evolution are as described in the above discussion:

- the identification of agents: e.g. foreman, warehouse worker, driver, office clerk;
- the conception for roles for these agents corresponding to their characteristic skills;
- the apportioning of responsibilities for particular phases within a given transaction;
- the refinement and formalisation of their precise observables and protocols.

In applying SPORE to developing warehouse requirements, this general process is emulated using computer-based technology. The ISM we construct for this purpose incorporates the seed ISMs for the warehouse; the form-based abstractions that capture the state of the BPM and the activities of the agents; 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 transformation from a paper-based to a computer-based ISM illustrates the potential of SPORE as a framework for business-process re-engineering.

The distributed nature of EDEN enables us to separate the viewpoints of the agents in the model, and to complement these with an external interpretation. In the first instance, computer-based forms are used to represent the environment for each agent's interaction. The mechanisms through which a particular kind of agent, such as a warehouse worker, interacts can be subsequently elaborated through the development

of special-purpose interfaces. In this way, the distributed ISM serves as a medium in which to identify and enact appropriate transactions, and to debug and refine these through collaborative interaction between the various participants.

Examples of how requirements can be addressed by SPORE in this way include:

- 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.

5. Conclusion

In this paper, a problem-solving framework for cultivating requirements in an interactive and situated manner has been presented and illustrated with a case study of a warehouse distribution system. This framework requires participants to construct and exploit computer models as artefacts in order to explore and integrate individual insights. By means of these networked artefacts in a distributed environment, participants can collaboratively interact with the domain and with each other to shape their shared understanding on the basis of their current contexts and available resources. As a result, requirements as solutions to the identified problems in the application domain are cultivated in the REP and embodied in computer models that complement the documentary record.

Our research suggests that the situated process of requirements engineering can be appropriately supported by the broad computational framework of Empirical Modelling where human and artificial agency can be integrated in a natural fashion.

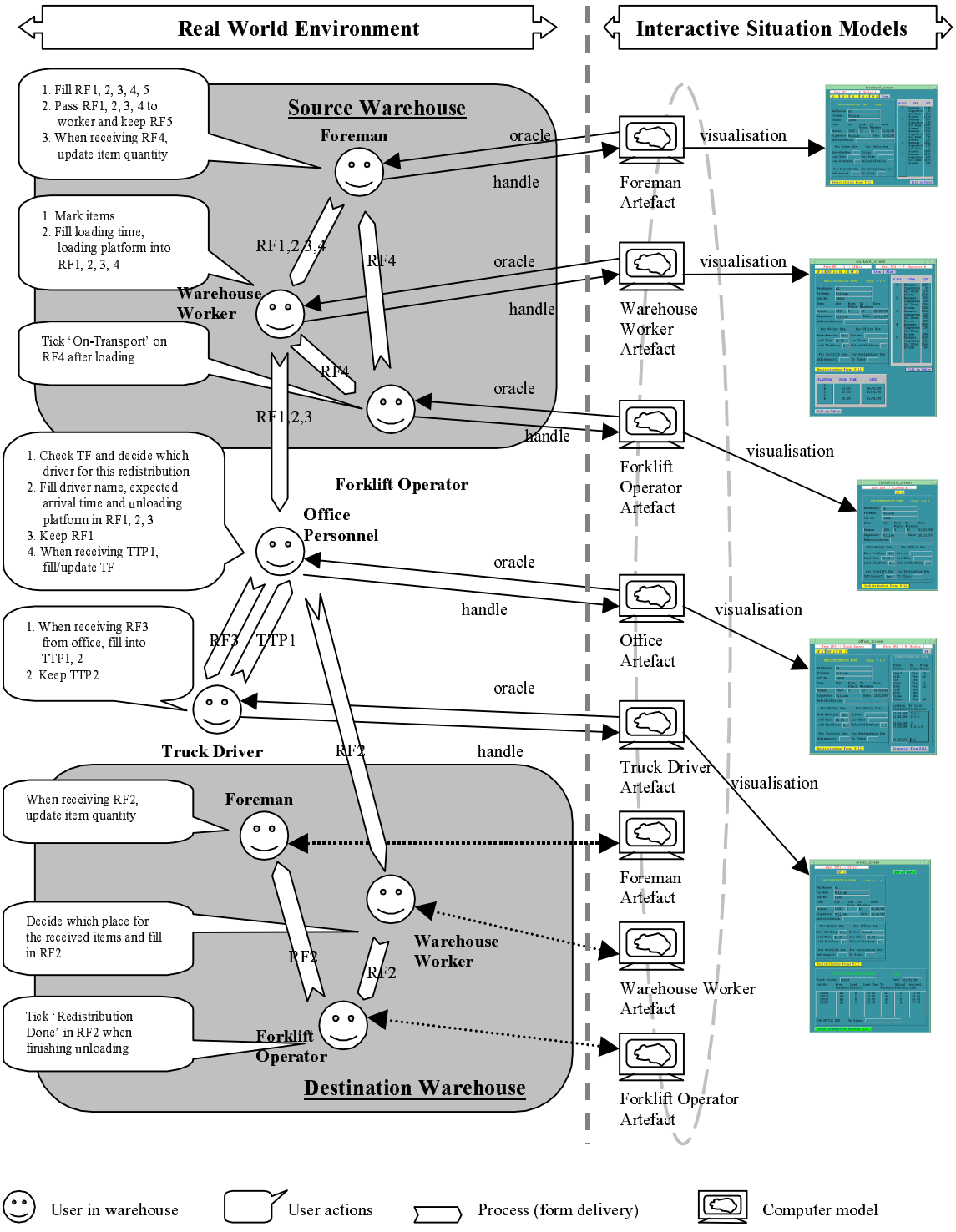

Figure 4: A collaborative working environment for manual redistribution between warehouses

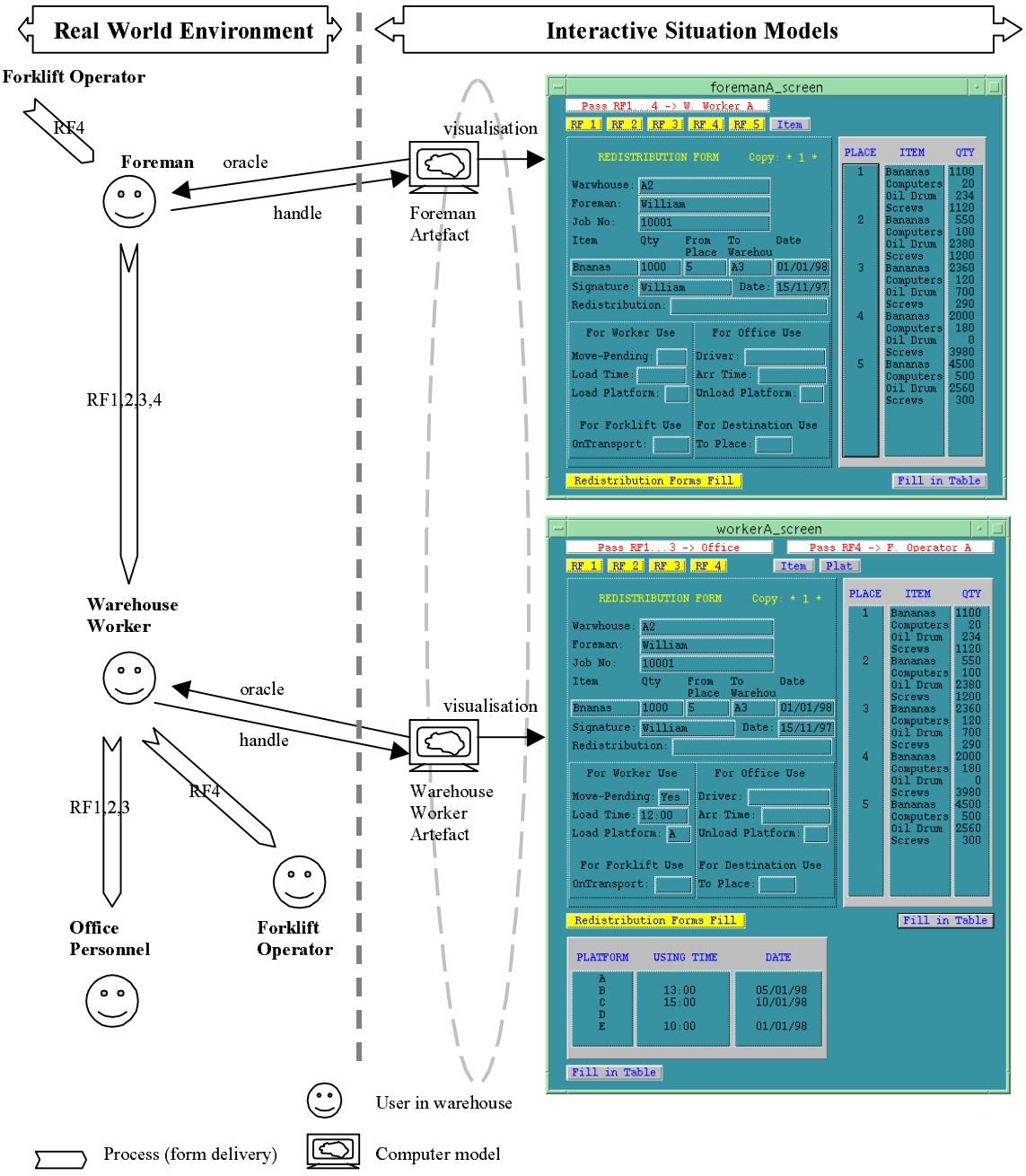

Figure 5a (above): Detailed view of the forms used in the warehouse artefacts

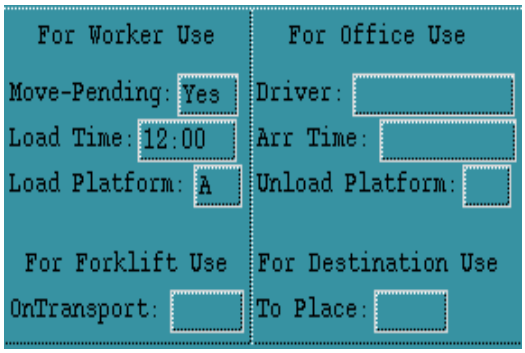

Figure 5b (right): Detail of panels representing observables (**handles** or **oracles**) for some warehouse agents

References

[BCDS93] A.J.C. Blyth, J. Chudge, J.E. Dobson and M.R. Strens. (1993) ORDIT: A new methodology to assist in the process of eliciting and modelling organisation requirements. Technical report No.456, Dept. of Comp. Sci., Univ. of Newcastle upon Tyne.

[Bey99] W. M. Beynon. (1999) Empirical modelling and the foundation of artificial intelligence. In C.Nehaniv (ed), *Springer LN in AI*.

[Blu93] B. I. Blum. (1993) Representing open requirements with a fragment-based specification, *IEEE trans. on Systems, Man, and Cybernetics*, V23(3), 724-736.

[Bub95] J.A. Bubenko Jr. (1995) Challenges in requirements engineering. In *Proc. of the 2nd Inter. Sym. on Requirements Engineering*, pp. 160-162.

[Cro94] C. Crook. (1994) *Computers and the collaborative experience of learning*. Routledge, London.

[Dav93] A. Davis. (1993) *Software requirements: objects, functions and states*. Prentice-Hill.

[DS97] S.E. Donaldson and S.G. Siegel. (1997) *Cultivating successful software development: a practioner's view*. Prentice Hall.

[Fin94] A. Finkelstein. (1994) Requirements engineering: a review and research agenda. In *Proc. 1st Asian & Pacific Software Engineering Conference*, IEEE Press, pp.10-19.

[Fis91] G. Fischer. (1991) The importance of models in making complex systems comprehensible. In M.J. Tauber and D. Ackermann (eds.), M*ental models and human-computer interaction 2*, North-Holland, pp. 3-36.

[Gog94] J.A. Goguen. (1994) Requirements engineering as the reconciliation of technical and social issues. In M. Jirotka and J. Goguen, editors, *Requirements engineering: social and technical issues*, Academic, pp.165-200.

[Gog96] J.A. Goguen. (1996) Formality and informality in requirements engineering. In *Proc. 2nd Inter. Conf. on Requirements Engineering*, pp 102-108.

[IEEE90] IEEE-Std. '610'. (1990) *IEEE standard glossary of software engineering terminology*. Institute of Electrical Electronics Engineers, New York.

[JCJO92] I. Jacobson, M. Christerson, P. Jonsson and G. Övergaard. Object-oriented software engineering. Addison-Wesley, 1992.

[JP94] M. Jarke and K. Pohl. (1994) Requirements engineering in the year 2001: On (virtually) managing a changing reality. *Software Engineering Journal*, V9(6), 257-266.

[KS98] G. Kotonya and I. Sommerville. (1998) *Requirements engineering: processes and techniques*, John Wiley & Sons.

[LK95] P. Loucopoulos and V. Karakostas. (1995) *System requirements engineering*. McGraw-Hill.

[Rus97] S. Russ. (1997) Empirical modelling: the computer as a modelling medium. *Computer Bulletin*, pp20-22, April.

[Rya95] K. Ryan. (1995) Let's have more experimentation in requirements engineering. In *Proc. of the 2nd Inter. Sym. on Requirements Engineering*, 1995, pp.66.

[SB82] W. Swartout and R. Balzer. (1982) On the inevitable intertwining of specification and design, *Communication of ACM*, V25(7), 438-440.

[Smi97] A. Smith. (1997) *Human computer factor: a study of users and information systems*, McGraw-Hill.

[SS96] J. Siddiqi and M.C. Shekaran. (1996) Requirements engineering: the emerging wisdom. *IEEE Software*, V13(3), 15-19.

[SS97] I. Sommerville and P. Sawyer. (1997) *Requirements engineering: A good practice guide*, John Wiley and Sons.

[STM95] P. Sallis, G. Tate, and S. MacDonell. (1995) *Software engineering: practice, management, and improvement*. Addison-Wesley.

[Suc87] L.A. Suchman. (1987) *Plans and situated actions: the problem of human-machine communication*. Cambridge University Press.