Empirical Modelling:
A New Approach for Understanding Requirements

Pi-hwa Sun, Meurig Beynon

Department of Computer Science

University of Warwick

sun@dcs.warwick.ac.uk, wmb@dcs.warwick.ac.uk

Abstract

Any task of understanding requirements involves interaction amongst all participants. This motivates the investigation of frameworks and techniques that can support and enrich human interaction. In this paper, three different relationships that can shape interaction are identified: subordinative, coordinative, and collaborative. Traditional patterns of interaction favour relationships of the first two kinds, since they presume a clearer separation between analysis, design and use. Whilst subordinative and/or coordinative relationships in interactions for eliciting requirements are valuable, we believe that they have been given disproportionate emphasis because of current techniques for knowledge representation and paradigms for computer-based modelling. Failure to support collaborative relationships is particularly significant in the exploratory phases of requirements analysis, when it is appropriate for the perspectives of designers, users and analysts to be equally influential. This paper proposes an effective computer-based approach to tackling many of the problems associated with this kind of collaboration.

1. Motivation and Background

Any task of understanding requirements needs to be carried out through interactions amongst all participants. The different relationships that can shape the interaction have provided the foundation for most models and methods in requirements engineering. There is a particularly significant distinction between coordinative and subordinative relationships. A coordinative relationship stresses the importance of user participation in design, and postulates responsibilities for all the participants. A subordinative relationship assumes that users should be able to provide all the knowledge required by designers because only they know what they want.

Traditional patterns of interaction favour relationships of these two kinds, since they presume a clearer separation between analysis, design and use than modern business practice and associated information technology promotes. In the development of

information systems, it is standard practice for feedback from users to affect the product. This feedback operates both in validating and debugging the original design, and in its subsequent enhancement. In concurrent engineering of other products, the use of information technology has subverted the rigid sequential stages of the traditional design process. The ease with which design representations can be visualised and modified enables wider and more opportunistic intervention from all kinds of participants. In these contexts, the interaction for understanding requirements becomes exceedingly subtle [SKVS95]. In effect, the design of a software system and the shaping of the requirements satisfying the need of users often needs be negotiated in a symbiotic fashion. The interaction amongst all participants that is appropriate in this context will be characterised as a collaborative relationship.

A useful analogy can be drawn between the relationships of all participants for understanding requirements and the ones of teacher and pupils in a classroom. A subordinative relationship resembles a lecture context, where the teacher imparts knowledge in the role of the expert, and there is no participation from the pupils. A coordinative relationship, in which a rigid agreement sets out the respective responsibilities of designers and users, resembles a tutorial context in which the teacher imparts knowledge through a prescribed pattern of small presentations, exercises for the pupils and evaluation of their performance. A collaborative relationship is concerned not only with responsibilities but also with expectations, beliefs and other psychological states that make understanding by learning more feasible and powerful [DL91]. The appropriate context for such interaction resembles a seminar, where the precise learning goals are not set out initially, and the knowledge content is shaped dynamically by the contributions of the participants. In the same way that all three paradigms can be used in one educational context, each of the three different kinds of relationship amongst all participants can be represented in the same process of understanding requirements.

Collaborative relationships are concerned with understanding that is socially distributed. They engage with issues of subjectivity and objectivity associated with distributed cognition [Hut95a, Hut95b] and common knowledge [Cro94, Edw87]. This involves a reappraisal of distinctions that are taken for granted in other contexts. There is potential for several kinds of conflation:

- between the roles of all participants,
- between the properties associated with individuals and with artefacts,
- between the characteristics to be attributed to the internal mind and to the external environment.

In a collaborative relationship, there is no possibility of relying entirely upon closed-world representations and preconceived patterns of interaction. The interaction amongst all participants has to be situated intelligent interaction that can only be planned in advance to a limited degree, and knowledge for understanding emerges on-the-fly.

There are several approaches to understanding requirements, but one of the most efficient and cost-effective ways is by computer-supported modelling. Conventional computer-based modelling is better oriented towards assisting subordinative and coordinative rather than collaborative relationships. As the above discussion indicates, it is essential to

develop modelling techniques that:

- allow data about requirements to be collected (as if) in such a way that participants are engaged in activities in their customary context [Gog96, LK95];
- make it possible to visualise and analyse activities from the viewpoints of different participants;
- provide for open-ended interaction.

The aim of this paper is to introduce a new approach - Empirical Modelling (EM) - to support collaborative interactions in a situated and distributed context. An exploratory study of a historic railway accident is used to highlight the application of EM in modelling for collaborative relationships. The implications of this investigation for future research are considered in the concluding section.

2. Empirical Modelling for Understanding Requirements

Empirical Modelling, as developed by the authors and their collaborators at the University of Warwick over several years, involves computer-supported modelling with an emphasis on learning [Bey98]. Its fundamental concepts are observable, dependency, agent and agency.

- An *observable* is a characteristic of the modelled environment to which an identity can be attributed. It can be physical or abstract in nature, e.g. the power of the engine, the position of the airplane, and the balance of a bank account, the time on the clock.
- A *dependency* represents an empirically established relationship between observables. It is not merely a constraint upon observables, but reflects how the act of changing one particular observable is perceived to change other observables predictably and indivisibly.
- An *agent* is an instigator of change to observables and dependencies. A passive agent will only act in a responsive mode, but an active agent may act autonomously.
- An *agency* represents a permission to access an observable. Agency can be associated with an observable whose presence is intermittent rather than persistent. So far two kinds of agency have been considered: permission to observe and permission to change.

EM is a powerful form of interactive modelling. It allows 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 mentioned above, and concurrently constructs a computer model that metaphorically exhibits similar patterns of observables, dependencies, agents and agencies. In this way the evolving insight of the modeller is reflected in coherence between an abstract 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 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 insight.

EM is a means of constructing knowledge in an experiential rather than a declarative fashion; the modeller's insight is expressed as 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 on 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 understanding requirements, especially for the purpose of re-engineering.

EM activities are carried out with reference to an external situation, even though in practice this situation can be imaginary rather than concrete. Practical experience of EM confirms its status as a situated modelling method, and activities in EM exhibit Goguen's "qualities of situatedness": emergence, contingence, locality, openness and vagueness [Gog94, Gog96] (cf. [Rol93, JP93]). The main reason why EM exhibits these qualities is that, because of the nature of the modeller's interaction, the process of formulating definitive scripts is never separated from the modelling context. At any stage, the modeller can modify or reinterpret the script so as to change this context. This can reflect a shift in viewpoint on the part of the modeller, a reappraisal of the external situation, or a refinement of the mode of observation. And though the syntax of a definitive script is formal, its semantics is established dynamically and informally through the modeller's interactions with the model and with the external situation. Moreover, automatic maintenance of dependencies amongst observables leads to techniques for retrospective revision, as commended by Goguen in his discussion of tools to support requirements engineering [Gog94].

A practical software tool, the dtkeden interpretor, has been implemented to support most of the concepts of EM. It provides an interactive interface through which each user can

construct definitive scripts to represent his/her personal construal of the external situation, and can observe graphical representation of the current construals of other users. The core part of dtkeden, the dependency maintainer, is a virtual machine and can automatically maintain the dependency of given definitive scripts. The current version of dtkeden provides a stand-alone and a distributed environment for carrying out EM. Many case studies have been developed to demonstrate the advantages and applications of EM (see our website: http://www.dcs.warwick.ac.uk/modelling).

EM is well-suited for application to understanding requirements. An immediate reason is that EM can support collaborative interactions in their situated and distributed context. In EM terms, all participants are agents. Real-world phenonema which they are observing individually are represented by constructing scripts that embody their private construals. In effect, the personal perspective and insight of each individual is recorded in an experiential manner with reference to an embodiment of their construal and their personal observation of the external world. In this way, all participants can construct their personal understanding of requirements throughout the evolution of the artefact in their own computer in an interactive, open-ended way. In other words, knowledge of requirements in EM is represented implicitly and metaphorically using an artefact, rather than expressed in an abstract specification. Moreover, the processes of creating and accessing this knowledge are associated with learning through experience of interacting with the artefact. This meets the need identified by Naur [Nau95] for incorporating experiential knowledge to complement knowledge as defined and processed by "logic and rules".

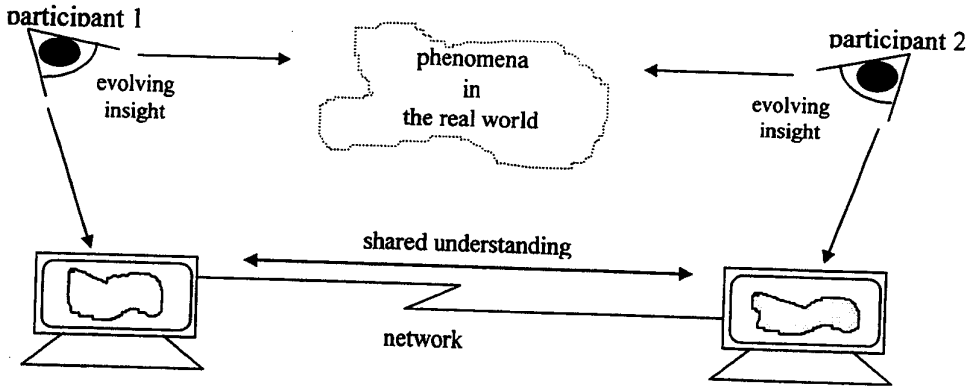


Fig. 1. Shared understanding between participants

In this context, understanding requirements amongst all participants becomes a collaborative task of building up shared understanding through these artefacts. As pictured in Figure 1, shared understanding amongst participants is socially distributed as it evolves through interaction between these embodied construals across a network. This process has a crucial role in introducing objectivity to the model, as it implicitly connects individual insights. More significantly, each participant can collaboratively contribute to the evolution of shared understanding. Generally speaking, greater consistency between the individual perspectives is associated with a better understanding of requirements. For this reason, participants continually refine their observations and experiments with a view to

achieving more coherence and consistency. This process is open-ended, and consistency can only be complete in relation to some restricted work activities and assumptions about reliability and commitment. In practice, there are always singular conditions under which a superior power must be invoked to mediate or arbitrate where there is conflict and inconsistency. Scope to exercise such discretion is a fundamental element of EM, as the modeller has the privileges of a super agent.

3. A Case Study: Shared Understanding in a Historical Railway Accident

A railway accident that occurred in the Clayton Tunnel near Brighton in 1861 [Rolt82] has been studied using EM and dtkeden. The modelling has involved constructing computer-based artefacts to represent the perspectives of four human agents involved in the accident, and to coordinate these from the point of reference of an external observer with exceptional state-changing privileges. One significant motivation for making such a model is to gain insight into the individual understanding of signalmen and drivers on their work practices at the time, and into how these may have contributed to the accident. Figure 2 shows a view of the Clayton Tunnel from the perspective of the signalman Killick.

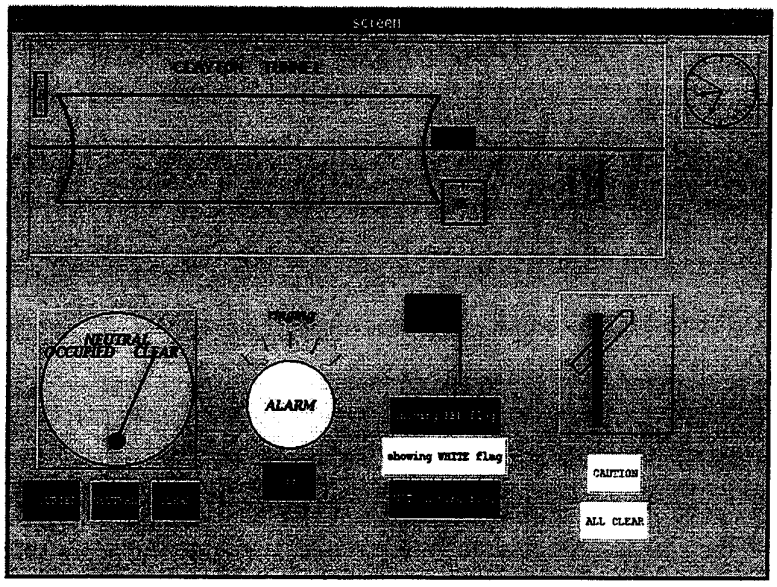


Figure 2. A signalman's view of the Clayton Tunnel

By way of illustration, consider how the two signalmen, Killick and Brown, whose location is indicated by the boxes labeled K and B in Figure 2, communicate with each other via the telegraph shown in the bottom-left corner of the figure. The private views of these two agents are as follows:

```
needlePosition is which_clicked;
which_clicked is (clear_clicked):(1):((neutral_clicked):(0):((occupied_clicked):(-1)));
```

signalman: Killick

```
needlePos is clicked;
clicked is (click_clear):(1):((click_neutral):(0):((click_occupied):(-1)));
```

signalman: Brown

Definitions of this nature specify computational artefacts to represent the personal agent perspectives. Different naming conventions for observables are used to reflect the independence of the agent's observation. Note that, for simplicity, the definitions concerned with visualization are omitted. The order in which the various definitions are introduced is not as important as in traditional programming because dtkeden automatically maintains dependencies amongst observables. For example, in the case of Brown, the value of the observable "needlePos" will be changed whenever the value of the observable "clicked" is changed. Further description is needed to express the way in which the signalmen inform each other about the state of the Tunnel. This could be as follows:

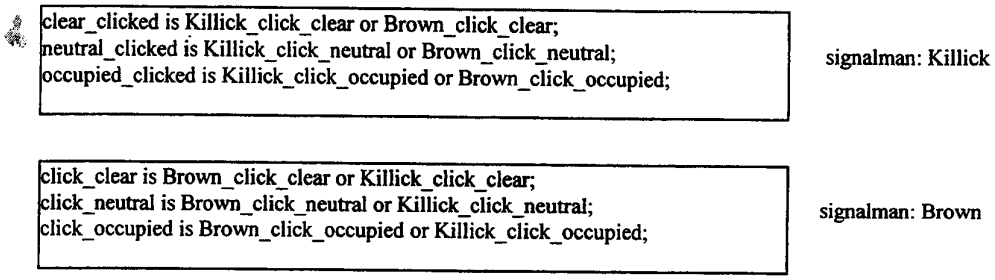
```
Func send1: Killick_click_clear{
  if (click_clear)
    sendAgent("Brown", "Killick_click_clear=TRUE;");
  else
    sendAgent("Brown", "Killick_click_clear=FALSE;");
}
Func send2: Killick_click_neutral{
  if (click_neutral)
    sendAgent("Brown", "Killick_click_neutral=TRUE;");
  else
    sendAgent("Brown", "Killick_click_neutral=FALSE;");
}
Func send3: Killick_click_occupied{
  if (click_occupied)
    sendAgent("Brown", "Killick_click_occupied=TRUE;");
  else
    sendAgent("Brown", "Killick_click_occupied=FALSE;");
}
```

signalman: Killick

```
Func sendA: Brown_click_clear{
  if (click_clear)
    sendAgent("Killick", "Brown_click_clear=TRUE;");
  else
    sendAgent("Killick", "Brown_click_clear=FALSE;");
}
Func sendB: Brown_click_neutral{
  if (click_neutral)
    sendAgent("Killick", "Brown_click_neutral=TRUE;");
  else
    sendAgent("Killick", "Brown_click_neutral=FALSE;");
}
Func sendC: Brown_click_occupied{
  if (click_occupied)
    sendAgent("Killick", "Brown_click_occupied=TRUE;");
  else
    sendAgent("Killick", "Brown_click_occupied=FALSE;");
}
```

signalman: Brown

Although the above mechanisms send definitions from one agent to the other, this does not of itself establish useful communication. To this end, there must also be a dependency between the definitions received by an agent and the private definitions within its own computational artefact. Only when definitions such as the following are added to their computational artefacts is shared understanding about communication using the telegraph reached:



This case-study illustrates how EM enables personal insight to be recorded in an open-ended experiential fashion, and how all participants collaboratively work together to reach a shared understanding through EM. It also highlights issues concerned with consistency of shared understanding, since a conflicting interpretation of events by a signalman and a driver was central to the accident. In addition, the case-study shows how "what if" interventions within **dtkeden** help to disclose information and reveal the need for additional data about the historical context being studied. Although this model is not directly concerned with understanding requirements, it does highlight the potential of EM in modelling for shared understanding amongst all participants.

4. Conclusion

Requirements are shaped by the goals and expectations of users, and the characteristics of the tools and processes that constrain their interaction. In so far as requirements follow intentional patterns, understanding requirements is intimately bound up with knowing what shared understanding has been established amongst all participants. Many techniques have been used in requirements engineering to develop and explore such shared understanding, such as brainstorming and interviewing. However, we believe that using the computer as a modelling tool is a particularly cost-effective and versatile approach. Constructing a computer-based prototype is one way in which the designer's conception can be validated by users. Prototyping restricts interaction with the computer model in preconceived ways, and is accordingly best suited for the exploration of coordinative relationships between designers and users. In contrast, EM is oriented towards a collaborative relationship, where the interaction among all participants is situated, distributed and open-ended. In such a framework, it is possible to explore diverse aspects of requirements that lie outside the preconceived framework established by design.

Our work is on-going and there is still much scope for improvement and future work. For example, the concept of a higher-order dependency has been introduced and a new dependency maintainer has recently been developed in Java. In addition, established techniques of requirements elicitation, such as scenario-analysis, may benefit from integration with EM.

References

[Bey98] W.M. Beynon. Empirical Modelling and the Foundations of Artificial Intelligence. In *Proc. CMAA*, ed. C. Nehaniv, University of Aizu, April 1998.

[Cro94] C. Crook. Computers and the Collaborative Experience of Learning. London:Routledge, 1994.

[DL91] L.Davies and P. Ledington. *Information in Action: Soft System Methodology*. MacMillan Education Ltd, 1991.

[Edw87] D. Edwards, and N. Mercer. *Common Knowledge*. London: Methuen, 1987.

[Gog94] J.A. Goguen. Requirements Engineering as the Reconciliation of Technical and Social Issues. In M. Jirotka and J. Goguen, editors, *Requirements Engineering: Social and Technical Issues*. pp165-200, Academic, 1994.

[Gog96] J.A. Goguen. Formality and Informality in Requirements Engineering. In Proc. 2nd Inter. Conf. on Requirements Engineering, pp 102-108, 1996.

[Hut95a] E. Hutchins. Cognition in the Wild. MIT Press, 1995.

[Hut95b] E. Hutchins. How a Cockpit Remembers Its Speeds. Cognitive Science, 19, pp 265-288, 1995

[LK95] P. Loucopoulos and V. Karakostas. *System Requirements Engineering*. McGraw-Hill, 1995.

[Nau95] P. Naur. Knowing and the Mystique of Logic and Rules. Kluwer Academic Publishers, 1995.

[Rol93] C. Rolland. Modelling the Requirements Engineering Process. In *3rd European-Japanese Seminar on Information Modelling and Knowledge Bases*, 1993.

[Rolt82] L.T.C. Rolt. *Red for Danger*. Pan Books, 4th edition, 1982.

[SKVS95] I. Sommerville, G. Kotonya, S. Viller and P. Sawyer. Process Viewpoint. In *Proc. 4th European Workshop on Software Process Technology*, The Netherlands, Apr. 1995.