CHAPTER FOUR

Empirical Modelling Principles

Empirical Modelling (EM) is an approach to human-centred and situated computer-based modelling which has been developed at the University of Warwick since 1983. The principal aim of this thesis is to investigate the suitability of EM as a framework for software development to meet the need of BPR. As the research of this thesis is based on the framework of EM, this chapter will give an overview of EM, illustrate the fundamental principles and concepts of EM, and show how our approach differs from the modelling methods traditionally used in software system development.

4.1 Introduction

Empirical Modelling is a situated computer-based approach to modelling. EM is based on an unconventional approach to analysing phenomena which is *observation-oriented* rather than object-oriented in nature. This analysis is directed at accounting for phenomena in terms of the state-changing activities of *agents*. Actions of agents and interactions among agents are expressed as changes to *observables*. The way in which a change to one observable affects changes to other observables in a manner that is conceptually indivisible is modelled as a *dependency*.

The concepts of agent, observable and dependency are fundamental to EM. To some degree, these concepts are represented in current computing practices, for instance in 'agent-based systems', 'observers in the Java package `java.util`' and in 'spreadsheets'. What differentiates EM from the ad

hoc use of concepts relating to agency, observation and dependency is the special nature of the relationships between the modeller, the computer model (as artefact) and its referent. Specifically, the artefact stands in a special relation to the modeller's understanding of the referent (his 'construal') as informed by observation and experiment.

The nature of EM invests the concepts of agent, observable and dependency with an ontological status that is unrecognised, if not unrepresented, in their use in other contexts. In this thesis, it will be convenient to regard modelling methods other than EM as essentially framed in a 'classical' ontological framework, and to refer to them as 'conventional modelling methods', even though they may make use of agency, observation and dependency in some form. This classification is uncontroversial when standard modelling techniques for software system development are concerned, since these do not lead to the identification and embodiment of agency and dependency in the manner characteristic of EM.

The term 'empirical' reflects the fact that the EM principles are based on observation and experiment. The emphasis of our approach is on modelling (parts of) the world through experience, i.e. modelling our experience (that of the subject domain) by means of other experiences (of the computer model), in contrast to conventional approaches of system development and modelling which model systems by preconceived abstraction. Actually the term 'empirical modelling' was adopted in 1995 to make such a distinction between conventional mathematical modelling and our experiential models based on definitive scripts.

When describing the concept of circumscription in previous chapter, we mentioned that traditional modelling approaches try to formalise the development process, as it was presumed that good solutions can be obtained by following rigid processes and abstract rules. But the activities under such circumscription, for example when the designer follows a formal method and follows such rigid protocols, are quite different from the activities in our everyday life – the *situated activities* (cf. Suchman, 1987; Hutchins, 1995; Varela, 1979; Varela et al., 1991). This situated activity is difficult to prescribe as it involves the human's dynamic interaction – or 'social interaction' following Goguen (1994; 1996) – with

the external environment in which some unpredictable events may appear and humans may deal with the same situation in different ways according to their knowledge, experience and tools available at the time. Such the activities involve social *situatedness* – they can only be understood in relation to the particular and concrete situation in which they occur; they acquire meaning through interpretation in that situation (Goguen, 1996). That is to say, the activities of our everyday life are context-dependent and human-centred. Thus the formalised activities in conventional modelling (in the closed-world paradigm) can only reflect the real-world activities to some extent, as they are dominated by algorithms and are independent of the context. For example, when we solve a problem in everyday life, we are conceiving the solutions based on our knowledge about the problem situation and our past experience, rather than following a general law. Sometimes we may make a plan for this, but such a plan is not acting in the same way as an algorithm, as Suchman (1987) describes, most plans are simply used as a 'resource' rather than a source of control. This is a motivation for EM: to provide another approach to modelling through the use of a computer as an open-ended artefact. The computer is used in EM as a instrument to facilitate cognitive activity. Through interacting with the computer model, the correspondences between the mental model of the modeller, the computer model and the referent can be established. Through this process the modeller's understanding of the subject can be enhanced and a computer-based model that can serve as a rapid prototype of the proposed system will be created based on the modeller's understanding of the subject rather than his imagination.

To give an overview of our modelling approach: EM is observation-oriented and state-based, that is, all variables in the model are considered to represent the current state of the observed referent. Also EM provides a visual metaphor with interfaces for direct manipulation by the modeller to represent the observables of the real-world subject. That is, the model is constructed as a metaphorical representation of the subject which is consistent with the modeller's construal. EM is also definition-based: it uses spreadsheet-like definitions to express the dependencies between observables, which cannot be achieved by conventionally circumscribed modelling.

4.2 The Essential Empirical Modelling Concepts

The focus of EM is on the construction of artefacts (the computer-based models) for interaction and experiment in a domain in which the user may not understand the subject well. It is observation- and experience-based as we can analyse the domain and the subject through the identification of the structure of our experience in that domain. Also the analysing and modelling of EM are experienced rather than preconceived which brings the computer modelling closer to users and also preserves the openness in the artefacts that reflects our experience in the real world (Russ, 1997). In this section the details of what EM is and how it works will be described.

4.2.1 General Concepts and Principles

There are three key concepts in EM: observables, dependencies between observables, and agents which are acting through introducing or changing observables and dependencies. The definitions and characteristics of these concepts can be summarised as:

- An *observable* represents a element of the domain which can be perceived and identified, and whose current value in the model corresponds to the state of the referent. An observable can be either physical or abstract. The observable can only have meaning when referring to the perception and interaction of an agent with a feature of the subject or the model (Ness, 1997).
- A *dependency* is a relationship between observables which may have different interpretations through different viewpoint of agents. Any change to the value of a particular observable will cause changes to the values of other observables (called the *dependants* of that observable) in a predictable way. The changes to the observable and its dependants are indivisible.
- An *agent* here is a state-changing entity which is a family of observables and is capable of changing the states of observables and dependencies[1]. The agent in EM refers not only to the human

1. Generally the change to the values of observables is due to the actions of agents.

being but also to any component of the subject which is deemed responsible for changes of state. The *agency* represents the agent's responsibility or privilege for the state change.

The first step of the EM modelling process is to analyse the domain and identify the observations (i.e. the observables) which are relevant to the modeller. It is essential that the observations here not only mean visual observations but also the class of scientific observations which can in principle be made (Farkas et al., 1993). For situated modelling, the presence of observables can also be intermittent rather than persistent (Sun, 1999), that is, the observables can appear or disappear at any time in response to the current situation. After analysing the observables, we can recognise the dependencies between them and define the agents associated with these observables. We group the observables around the agents which act as sources of change in those observables. The appropriate classification of observables into agents is essential in the modelling process as it decides the behaviour of agents and the overall effect upon the state of changing any observable.

The shaping of change in the observables is through dependencies which are expressed by definitions. The dependencies in EM reflect how the change of the values of one observable will predictably and indivisibly change the values of its dependants. Chen et al. (2000a) identify characteristic EM dependencies as 'law-like' dependencies which operate either like the physical constraints such as Hooke's Law or Newton's Law or the abstract conventions of when a game has been won. Further the dependencies between observables can also invoke a *hierarchical* kind of state change as they can 'propagate' the effect of changing one observable to all relevant observables directly and indirectly dependent on it (Sun, 1999). All the observables, dependencies and agents identified are provisional and subjective as they reflect the viewpoint of the modeller. That is, the identification of agency and dependency is based on previous knowledge and past experience, as well as new observation and experiment.

Definitive Representation of States

The key theme of EM is to construct a computer-based artefact to represent a subject-domain based on the concepts mentioned above in a situated and open-ended manner. In EM the definitive notation as a simple programming notation provides the support to do this. The process of definitive representation of states includes representing observables by *variables* and introducing *definitions* to express the dependencies between observables. A definition takes the form:

```
q is f(x, y, z)
```

where `f(x,y,z)` is a formula similar to the formula used to assign a value to a variable in conventional programming languages. Here `q`, `x`, `y` and `z` are variables which represent observables in the situation. The variables `x`, `y` and `z` are defined elsewhere in the script and `f` is a operation (usually a user-defined function) which reflects the perceived dependency of `q` on `x`, `y`, `z`. The definition acts as a one-way dependency rather than a constraint. The values of variables 'metaphorically' represent the observed situation from the viewpoint of the modeller, rather than the mathematical abstractions of numbers in the conventional sense. The current values of variables should conform to the current observations of the corresponding observables in the real world. The value of the variable `q` (the dependant) is always equal to the evaluation of the function `f` with the current values of the variables `x`, `y` and `z` (the dependees)[2]. Any change of the value in a dependee will lead to a re-evaluation of the value of its dependant. The artefact constructed in an EM framework will have a collection of observables, and each of these observables has its own definition to represent the dependency. A collection of these definitions – a *definitive script* – corresponds to a single state of the model. Any state of the model should correspond to a state of its *referent* in the situation. The order of definitions in a script is not important as any redefinition of a variable will automatically re-evaluate all its dependents and bring them to a new state. This feature allows the modeller to be more relaxed and have more freedom in planning and developing the script than when writing a conventional program. The structure of the EM model based on the definitive representation of state can be summarised in Figure 4.1.

2. The '**is**' in the definition indicates that the dependency is maintained automatically.

The semantics of a definitive script is similar to that in a spreadsheet in which a user identifies the relationships between cells and any change to the value of one cell will automatically cause other dependent cells to update. It represents one state in the real world which can be immediately experienced (Beynon and Russ, 1994). Modelling such immediate experience in the EM manner makes our approach different from other programming methods as the latter are based on more abstract ways of describing the circumscribed behaviour of the proposed system. The main idea of spreadsheets – state change through dependency and agency – has not been made central in conventional approaches to programming (Beynon et al., 2002). In EM, the typical interaction of a modeller is to modify a definitive script by redefining a variable. The activity of on-line 're-programming' the model resembles 'what-if' activity in the spreadsheet. Also, as the values of variables in the script represent the observations of the referent in a particular state, our model acts in a similar role to that of an engineering prototype. Thus our approach is 'experience-based' because the models are continuously open for the modeller to re-program based on his new experience of the world as well as his experience of the current state of

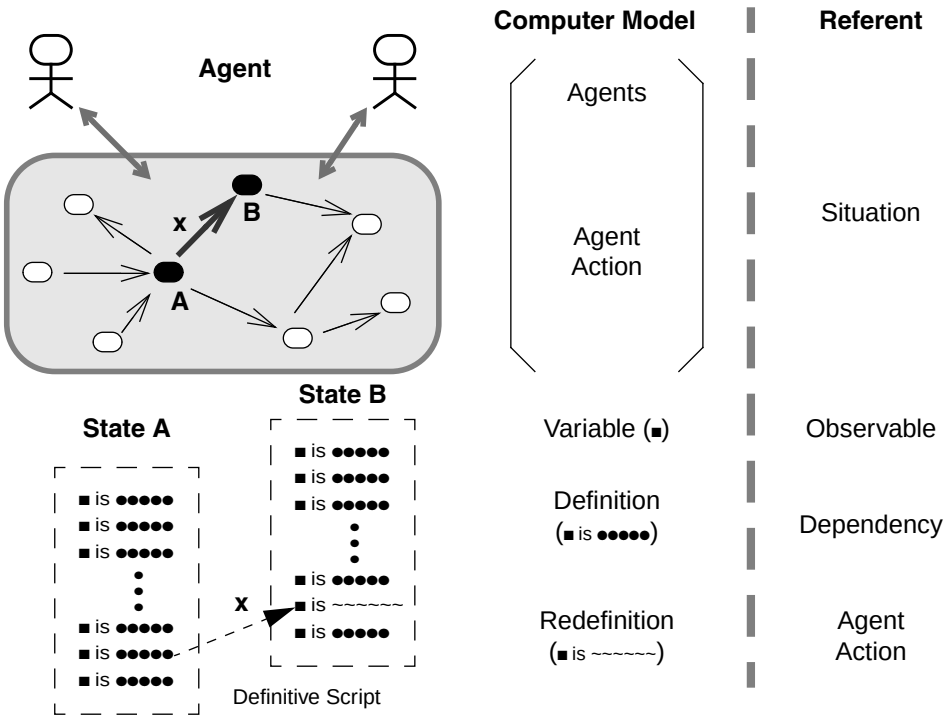

Figure 4.1 The Framework for the Modelling

the model. This can explain the primary focus of EM on state[3] in the sense of *state-as-perceived-by-agent* or *state-as-experienced* rather than the more conventionally abstract and limited sense (Chen et al., 2000a).

4.2.2 Software Tools for Empirical Modelling

The aim of EM is to enable the modeller to extend and experience his mental model of the real-world situation through the interaction with the computer-based model. The EM concepts mentioned above are supported by several tools which have been developed by this group.

The construction of EM models is based on the concepts of observables, dependencies and agents which are used to represent the modeller's viewpoints and interpretation of the domain, i.e. his experience of the domain. The modeller's beliefs and perceptions are then recorded and organised using a open-ended notation: *LSD* (Language for Specification and Description). The purpose of LSD account is to describe a phenomenon by identifying the observables which are associated with an agent's interactions, as well as the roles and potential actions of agents in terms of such observables. Actually some observables may represent the states which cannot be directly perceived by the modeller or include the states of mind which are not directly accessible (Beynon et al., 1996). The semantics of LSD is different to that of conventional notations and because of this the artefact we construct represents the states which are states-as-experienced.

To model the behaviour of agents, the first step is to describe those observables whose existence depends on that of an agent (*states*), those that can be conditionally redefined by the agent (*handles*), those whose values act as stimuli for agent's response (*oracles*), the dependencies between observables (i.e. the stimulus-response patterns) perceived by the agent (*derivates*[4]), and the privileges of agents to perform state-changing action (*protocols*). We record these observables in LSD which indi-

3. The term 'state' used in EM is used very broadly to include the state of mind in subject experience, the state of a computer artefact and the state of its real-world referent.
4. Sometimes a derivate may also represent an idealised relationship.

cates both the internal perception of agents as well as the external perspective of the modeller based on his personal construal of subjects. The identification and classification of such agents, agency and observables in LSD reflect the modeller's observation of both the model and its referent. Further, an LSD account generally allows different operational interpretations, which correspond to different presumptions about the environment for the interaction of agents, and the nature and reliability of their stimuli-response actions (Ness, 1997). The notation of LSD is informal but structured which is helpful throughout the construction process (like the role of use case models in the OOSE modelling process). LSD serves different purpose from conventional specifications as its interpretation is personal and its establishes a 'soft' boundary to be continuously developed, whereas the conventional specification is the 'culmination' of earlier drafts that establishes a hard boundary which is not intended for revision (Beynon et al., 2002).

The LSD accounts can be animated through the use of the Abstract Definitive Machine (*ADM*). As the redefinitions in a definitive script can represent the actions of different agents, in ADM the state transitions are represented by the parallel redefinition of variables in such a script. The ADM is the environment for the modeller to act as a superagent (in the role of arbitrator) to add appropriate scenarios for the actions and interactions of agents. In this sense the degree of agent autonomy in the simulation depends on the extent to which the ADM program is driven by human intervention rather than the automatic controls of perceived protocols (Beynon, 1994). Generally the modeller manually transforms an LSD account into an executable ADM program. Through the animation the interaction between the LSD agents can have operational meaning and the modeller has an unrestricted power to redefine the existing definitions or add new definitions. This means the patterns of interaction between agents within the artefact is entirely directed by the modeller (the superagent). Through this on-the-fly interaction with the model, the modeller can improve his understanding, and ensure the system behaviour modelled in terms of corporate behaviour of a set of agents is meaningful with reference to his external observation.

The tool *EDEN* (the Evaluator for Definitive Notations) is another interactive tool for realising and exploring the state changes consistent with the definitive script. EDEN is developed on the basis of the principles of EM and has so far been found to be the most successful tool for EM. Within EDEN there is

a window-based interface (based on the Tcl/Tk interpreter) which provides an interactive environment through which the modeller can at any time (1) create a new variable or modify the value of an existing one; (2) create dependencies by introducing new definitions or redefining existing ones; (3) observe the current values of variables and their definitions as well as the influence due to the modeller's intervention. There are two main features of EDEN: the scripts of definitions on variables, and stimulus-response actions triggered by changes to the values of variables in the scripts. There is a close correspondence between the LSD account and the EDEN animation: the derivates determine the definitive script and the script defines the state of the model; and actions are used to implement protocols. The visualisation of the model is realised using the definitive notations *DoNaLD*[5] (for the two-dimensional line drawing) and *Scout*[6] (for window layout) using the **`tkeden`** interpreter. There are three main windows in the screen display of EDEN (Beynon et al., 2000c):

- The EDEN *input* window: Through this the values of variables and their definitions can be modified and new definitions, functions and actions can be created to change the current state of the interactive situation model (ISM). The modeller can interact with the ISM by typing definitions line-by-line which represents the introduction of new observables and the redefinition of existing ones.
- The *interface* window: This window provides an interface for observations, visual feedback and interaction by the mouse. It can serve different functions according to the configuration of the script.
- The *commentary* window: This shows information about the current state of the ISM and the outputs of EDEN actions or requests by the modeller.

Within EDEN, **`tkeden`** is the interpreter to maintain the dependencies between all kinds of the internal data values as well as the dependencies between the values and other elements (geometric, textual or iconic) on the screen. As in spreadsheets, any change to the value of a variable will cause **`tkeden`** to automatically change all values that depend on that variable and re-evaluate their values. Through this

5. The **D**efinitive **N**otation for **L**ine **D**rawing.
6. The notation for **SC**reen Lay**OUT**.

the dependencies are maintained as indivisible and propagated state changes which reflect certain state changes of the referent. The change to the values of variables can be achieved by either textual inputs through the input window or graphical inputs via the interface window. The construction of an ISM via the EDEN notation is situated rather than preconceived as no one can predict what definitions will be added or redefined by the user.

The environment of EDEN offers some limited assistance to the user for the management and debugging of scripts (Chen et al., 2000a). The notations of EDEN are mainly a tool for research purposes and are sufficient to explore the principles of EM. In this situation EDEN lacks the features of robustness, efficiency and consistency needed for large-scale applications for commerce or industry.

4.3 Modelling Activities in EM

EM is a form of interactive modelling which allows the modeller to construct an artefact through the use of computers. In the construction of, and interaction with, the EM model, the modeller can 'embed' the knowledge about observables, dependencies and agencies in such a computer-based model (Beynon et al., 1996; Beynon et al., 2000c; Ness, 1997; Sun, 1999) and gain a better understanding about the domain in which the system will be developed as well as the real-world situation.

4.3.1 The Modelling Process under the Definitive Notations

The modelling of EM and the construction of an ISM is summarised in Figure 4.2. The modeller must firstly build up a correspondence between the EM model (i.e. the ISM) and its referent. This process typically involves the identification of dependencies and the introduction of definitions into an ISM. The modeller here is broad including the role of an system user. In constructing the correspondence, the modeller initially identifies relevant agents, observations and protocols to form the partial LSD account. The LSD account can then be animated by the ADM or EDEN tool in which the definitions in the definitive script can be evaluated. The visualisation of the model can be established by DoNaLD and Scout

which represents the current state of such model. The modeller can intervene in the current behaviour of an ISM by modifying the values of variables or the dependencies and immediately experience the result. During the construction process, the modeller can make experiments and observations on both the model and its referent.

The association between the model and its referent consists of the correspondence between the states of the ISM and the states of the real-world referent (i.e. between the variables of an ISM and the observables of the subject). This correspondence is established by two correspondences: the one connects the modeller's mental model of the current situation (as expressed by knowledge of agency and

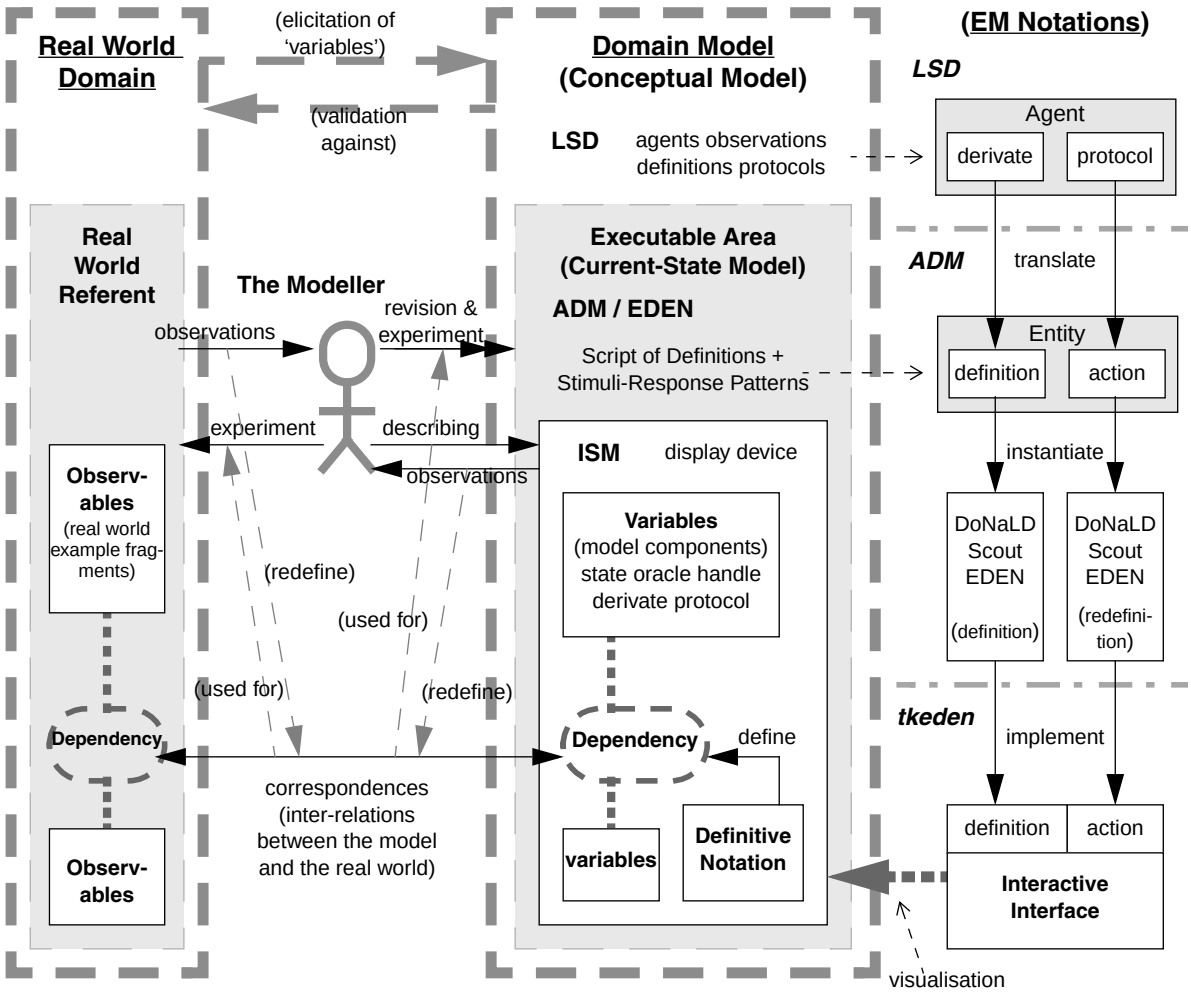

Figure 4.2 Empirical Modelling and its Tools and Notations

dependency, and expectations about interaction) with the ISM and the other one connects the mental model with its referent. These two correspondences are established by interactive activities such as the observations and experiments by the modeller. Any changes occurring in the model or the referent may affect the mental model of the modeller. The modeller can simulate different types of interactions with the real world by interacting with the ISM. Through modifying the dependencies in the model either by adding new definitions or refining the existing ones, the modeller can ensure and maintain the correspondence between the dependency in the model and the dependency in the external referent. The process of EM modelling enables the modeller to experiment with the effect of adding new elements into the system, experience parts of the system, and interact with the system in a unrestricted manner. The insight of the modeller gained through establishing such correspondence is expressed in coherence between his abstract explanatory model (the construal) in the mind and the physical embodiment of such construal in the ISM as well as the situation of the referent (Sun, 1999). In this way the mental model of the modeller is not only enriched but also embodied through the continuous and open-ended interactions with the ISM.

4.3.2 Model Construction

The central idea of EM is constructing models called *interactive situation models* (ISMs)[7] to supply the modeller with patterns of interaction which resemble the interaction and observation when experimenting with the referent. An ISM is open for the modeller to experiment with in a way similar to that in which the subject of the real world is open to experiment. The key idea is that the ISM is a model of the relationship between a situation and the observer (Beynon et al., 1998), which leads to a better human engagement in the interaction. We can introduce changes to the model and have direct experience of the results. In this sense any modification in the ISM can reflect a change in the real-world situation or the new knowledge gained or experiment made by the modeller. The interactions and the interpretation of states are uncircumscribed in an ISM because they are highly dependent on the situation as well as

7. The term ‘interactive situation model’ is introduced in this group as a synonym for ‘model generated using EM principles and tools’ (Beynon et al., 1998).

the knowledge and motivation of the modeller. That is, the construction of, and interaction with, an ISM take place in a situated manner rather than in the abstract manner of conventional approaches. The word 'situation' in 'interactive situation model' refers to this fact that the model is rooted in a concrete context that affects the modeller's expectation and interpretation (Chen et al., 2000a). The character of an ISM in this respect is similar to a spreadsheet where we may: (1) update the cells to reflect changes in the external situation; or (2) redefine the formula to reflect a new insight in the situation; or (3) introduce new cells and formulae for additional information or expectation about the situation. Both the EM model and conventional spreadsheet models are characterised by Chen et al. (2000a) as examples of ISMs because they incorporate the agency and dependency which are revealed during the analysis and construction processes. At the same time, the correspondence between the variables in a definitive script and the observables perceived by the modeller is maintained. To establish and maintain a correspondence between the real-world observables and the variables defined in the model is the essential principle of EM. This is achieved through the modeller's interaction with both the model and its referent in parallel. The process of establishing such a correspondence is iterative which involves continuously refining the model and making observations and experiments within the model and the referent. The aim of the correspondence setting is "to achieve consistency between the way in which sets of observables are indivisibly linked in change in the subject and the way in which the corresponding sets of observables are indivisibly linked in change in the model" (Ness, 1997).

The changes of state in ISMs occur through either the introduction of a new definition or the modification of an existing definition. A definitive script contains many definitions which represent indivisible dependencies. From a single redefinition may arise hundreds of automatic updates to other variables due to the relevant definitions. In the ISM, the state transitions are associated with agent action which is typically represented as a group of redefinitions to be executed in parallel. The modeller can act in the role of super-agent who can always intervene in any state by introducing definitions in an unrestricted manner. For this reason, the human and the automated activities in the ISM can be closely integrated.

There is no fixed and preconceived process of analysing a domain and constructing ISMs. That is, the modelling activity is open-ended and the modeller has flexible control over the redefinitions of the

model. Generally the changes of state in the ISM should reflect the expectations of the modeller but in reality observations may agree or disagree with expectations and therefore affect further construction. Usually an ISM presents only a partial and provisional artefact to the modeller (Chen et al., 2000a). This means the construction of EM models occurs in an experimental and incremental manner which reflects the modeller's knowledge of the domain. This process is also situated because the model reflects the personal experience and subjective viewpoint of the modeller by comparisons between experience with the domain and experience of interaction with the model. The distinctive characteristics of ISMs can be summarised as follows (Beynon et al., 1998):

- An ISM captures knowledge that is not task-specific. That is, the purpose of an ISM is to provide computer support for the modeller's conception of the referent and capture the knowledge about the referent by some metaphorical representation. Such knowledge here can broadly refer to anything the modeller can observe through interaction with the referent. It can be either the modeller's naïve observation about the current state of the referent or the modeller's expectation about its responses.
- An ISM represents subjective knowledge. That is, an ISM reflects the perception and experience of a particular person to a particular referent within a particular situation. Different people may have different viewpoints of the same model and sometimes these viewpoints are difficult to combine in system development due to their different concerns. For example, the user, analyst and implementer of a system have different application-specific knowledge. How an ISM can be the general framework for the representation and use of these differing perceptions is detailed in (Beynon and Russ, 1994).
- An ISM represents knowledge about interaction in particular contexts. That is, an ISM generates a direct metaphorical representation of a particular state of the referent. By interacting with an ISM the modeller can get immediate feedback of the consequences of some state-changing actions. This differs from conventional computer models as they represent objective knowledge about the intended functionality of the system and modes of user interaction for a specific goal.

Computer-Based Model as Artefact

The essence of an artefact is that it should imitate the interaction with the referent in the real world. Simon (1996) gives the definition of artefact in terms of form (inner environment), context (outer environment) and purpose:

> An artefact can be thought of as a meeting point – an 'interface' in today's terms – between an 'inner' environment, the substance and organisation of the artefact itself, and an 'outer' environment, the surroundings in which it operates. If the inner environment is appropriate to the outer environment, or vice versa, the artefact will serve its intended purpose. (p. 6)

According to this definition, an artefact is a kind of abstract boundary between the inner and outer environments, i.e. the form and the context. However such an abstract interface does not exist in the real world, it is represented in our mind or by artefacts which are constructed to represent knowledge (Ness, 1997). In EM we broadly define an artefact (or cognitive artefact) as an object or environment which is constructed for interaction which imitates interaction with the real-world system (Beynon and Cartwright, 1995). This means that an artefact should exhibit different states which correspond to the external states in the real world. Also it should provide an environment for modeller to interact in an open-ended manner and explore the correspondence between the artefact and its referent. Thus two kinds of artefacts in the design process are (Beynon and Cartwright, 1995; Cartwright, 1998):

- Artefacts with which the designer can interact in order to develop an insight into the nature and current status of an object being designed;
- Artefacts that inform the designer about the current status and progress of the design process.

One theme of our research is to find out how the computer can best be exploited in the construction of artefacts. Following a convention that is often useful in EM, the term 'computer' here refers to any reliable, interpretable, state-changing device (Beynon et al., 2002). The environment in which a computer operates is important for the interpretation of computer behaviour. This is the reason we refer to the construction of artefacts as *modelling* rather than conventional programming as the process includes

not only the development of algorithms and scripts but also the configuration of human agents and other resources within the system. The computer in the EM modelling process serves as a physical instrument with which the modeller interacts. This differs from the conventional way in which the computer acts as a means to implement an abstract algorithm or computation. In this respect, Beynon et al. (2001) suggest that the experience of constructing ISMs is close to the characteristic of an *instrument* as 'maintaining a relationship between aspects of states'. This is in contrast to the role of the computer in traditional software development as a *tool* which is developed to serve a particular purpose. The characteristic difference between instruments and tools also reveals the different emphasis of artefact use: in the tool-like use of artefacts, the concern is with the efficient and reliable progress to the goals; whereas in instrument-like use of artefacts, the focus is on the experience rather than the achievement of the goal. For this reason, constructing an ISM can be regarded as using an interactive instrument, as the construction is a situated activity that can be developed in an open-ended manner which involves exploration and experiment.

Ness (1997) shows in his thesis that the characters of EM and conventional system development are determined by the nature of artefacts used. That is, the nature of modelling activities may be different according to the kind of artefacts used by modellers. The following summarises some aspects of modelling activity that may be influenced by the nature of artefacts.

Artefacts The artefacts are used in system development to represent the structure and functionality of the proposed systems. The artefacts of EM correspond to the subject; but in conventional methods the artefacts correspond to abstract conceptual models of the subject. For example in OOSE (Jacobson et al., 1992), the use case model is constructed to specify the functionality of the system. The analysis model and design model are both used to represent the structure of the system: the former focuses on the logical structure of the system (under the ideal conditions) in terms of objects, whereas the latter refines and formalises the analysis model to meet the actual implementation environment. In contrast, the LSD account and the animation of an ISM resulting from **`tkeden`** interpreting the EDEN and DoNaLD scripts form the artefact during the modelling process of EM. The LSD reflects the observables and agents which are perceived by the modeller in the subject. Such representation ena-

bles the modeller to construct the model in an intuitive way without detailed knowledge of the structure and function of the proposed system (Ness, 1997). The modeller can observe and experience the function of the system by interacting with the model. The animation of an ISM represents the function of the systems and the users can animate the artefact by adding new definitions or redefining the definitions in the scripts in an experimental way.

Subjects It is argued by Ness (1997) that whether the subject (i.e. the system to be modelled) is novel or familiar to the modeller will essentially influence the suitability of a particular kind of artefact for representing the subject as well as the activities in the modelling process. The ISM seems to be more suitable for representing subjects which are novel to the modeller. As the ISM is a computer-based model of situations, it can be used to deal with many different subjects and Sun (1999) infers that such subjects are typically an 'intelligence-intensive' process (or a soft process) such as software system development or requirements understanding. By way of contrast, the artefacts in conventional modelling methodologies are more successful in representing the subjects more familiar to the designer.

Actions The actions in conventional system development are transformational in nature, whereas in EM the actions are generative and exploratory in nature (Ness, 1997). For example in object-oriented methods, the designers begin by identifying the roles of the proposed system and the correspondence between models and the subject is preconceived through imagination. It is transformational because the construction of models is transformed from the completion of the models in previous stages, for example, transforming the nouns or verbs in the requirements specification (or the use cases) into the analysis model and the transition from OOA to OOD. But as we showed in section 2.2, there are difficulties when integrating these models by applying the concept of objects. In addition, the designer may need to spend more and more time to modify the models whenever inadequacies are found in the artefacts. The actions in EM are generative and exploratory because the modeller typically generates the artefact firstly and then evaluates the artefact with the subject. Ness (1997) describes how the refinements (i.e. adding new definitions or redefining existing ones in a definitive script) reflect a better understanding of the subject by the modeller. The validity of the model can be ensured as its representation can directly correspond to the subject perceived by the modeller through observation and experiment.

Knowledge We have summarised two processes involved in the knowledge manipulation for system development in section 2.3: knowledge construction and knowledge representation. Conventionally the knowledge used by designers in system development comes from either their long-term memory or the experience of interacting with the subject. Thus most modelling methodologies focus on knowledge representation and their artefacts are established mainly to record such abstract conceptual mental models in a 'context-free' manner. In contrast, the focus of EM is knowledge construction[8] in which the modeller constructs their knowledge in a situated manner. The ISM is an interactive and open-ended artefact which not only represents but also enriches the modeller's knowledge (his construal) by knowing-by-doing.

4.3.3 The Comparison with Other Modelling Methods

The nature of modelling activities may be different according to the nature of the artefacts used. This is because the way the modeller interacts with the state of the model in EM is different from how this happens in conventional modelling methods. It is common in many conventional mathematical models that features of the application domain are represented early in the process by abstract objects and operations. This means it is necessary to preconceive the possible future states for the model and the proposed system. For example using object-oriented methods, the objects, containing attributes and methods, need to be defined and carefully planned in advance before used in program. These attributes form the *data* for manipulation in the computer model and such data will determine the complete state space (i.e. the operational domain) of the system and all possible interactions with it. Such a program can be edited and re-compiled but it still remains an inherent feature of the paradigm that the program forms a 'boundary' within which the state and interaction must be preconceived (Chen et al., 2000a). The mathematical model expressed in the programming cannot be formulated without knowledge of this boundary. The human interpretation of program state is therefore restricted by this boundary and involves an association between real-world observations and the abstractions of this program. It is obvi-

8. Thus the main modelling activity of EM is the construction of the model (the ISM).

ous that the computer's manipulation of data is affected by such interpretation and major problems may occur when exceptions and errors appear.

In EM the description of state is by 'observables' through which the state of the EM model is presented to the modeller by a perceptual process as the way in which we typically perceive and interact with the real world. It is suggested by Chen et al. (2000a) that there exists a *comparability* and *connectedness* between observations of the model and observations of its referent. Such comparability is absent in conventional modelling methods because there "the 'observation' of the model is the reading of an abstracted value, or the preconceived interpretation of a pre-programmed display". That is, even though observations can be performed in conventional modelling, such observations are intended to identify elements (such as objects or classes in object-orientation) whose nature is strongly associated with a particular situation (the ideal situation). These elements are preconceived and context-independent (i.e. isolated from the modelling process), and observation is described in terms of them. In contrast, the observation in EM is a subjective and situated experience which refers to the modeller's ability to apprehend the features of a particular situation directly (Sun, 1999). Without this key feature of observation, the modelling activities in EM will not be differentiated from conventional modelling, where the modeller relies on imagination or conjecture whilst lacking both comparability and connectedness.

The Key Features of EM Modelling Process

The activity of observation plays a significant role in the EM modelling process as through observation, the modeller can not only construct the current situation but also enrich his knowledge and understanding for dealing with future situations. We argue that there exist comparability and connectedness between the observation of the EM model and the observation of the referent. There are three main features of the EM model that can illustrate this issue. Firstly, the correspondence between the observable of the referent and the variable in a definition is direct and simple. By this, the referent can be metaphorically represented by the model. This is in contrast to many conventional programs in which the interpretation of variables and their states can be highly abstract and problematic. However the variables in a definitive script still have the 'abstracted' quality connected with conventional programs. But we view

this as a 'limited abstraction' in which the connection between the concrete and the abstract is deliberate and familiar just like the use of natural language for describing the world (Chen et al., 2000a). Of course from the modeller's viewpoint, a definition is functioning as recording a dependency between the observables. But in the other hand, from a computational perspective such a definition has an abstract semantics similar to a formula in a spreadsheet.

The second feature is that the observation of the model can be performed through a visualisation implemented in definitive notations which is indivisibly linked to the script of the model. This visualisation is implemented using the **`tkeden`** interpreter which maintains the dependencies in spreadsheet-like manner. This ensures the visualisation of the model metaphorically represents its subject and provides a better virtual environment for interaction and rapid prototyping. This is so-called 'readiness-to-hand'[9] by Winograd and Flores (1987) – because such definitive scripts are within the 'definitive world' – which provides the right coupling between the modeller and his action in the relevant domain. Chen et al. (2000a) describe the nature of such visualisation as implicit: "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 third feature, which combines the directness of the first two features, is the quality of interaction offered by the EM model. There is no preconceived constraint for the modeller to interact with an EM model and revisions can be made at any time. Like a experimenter, the modeller may not be aware of, or predict, what action will affect the states of the subject. Also there are some consequent changes which may automatically arise due to dependencies rather than from human actions. The interactions of a modeller with the referent may be different because his knowledge of the observables and their dependencies can be changed. In system development, we find that the essential need for interaction between users and the system makes it difficult to develop a system under a closed-world paradigm. This is because experience is involved in the interaction between users and the systems. A user may want to interact with the system in ways beyond the scope prescribed by its designers. This is similar to the situation of driving a vehicle when the purpose of the interaction is to explore the engine's full poten-

9. 'Readiness-to-hand' means one object is a part of the background which is taken for granted without explicit recognition as an object. They give an example that to a person doing hammering, the hammer does not exist because it is a part of the hammerer's world.

tial or experiment with new driving skills. Prescribing the functionality of the system, or the interactions between users and the system, will inevitably restrict the experimental opportunities and openness which is available for the users. As the EM model acts as an instrument, the interactions offered in an EM model allow the modeller to introduce and test new scenarios on-the-fly and support 'what-if' experimentation. When several modellers involved, this guarantees a close and continuous engagement among participants.

Where interaction with the state of models is concerned, one fundamental difference between EM and conventional modelling is the way the changes of state take place. With a conventional approach, the modeller must plan and preconceive change methods, i.e. the inputs and outputs before model construction in order to prescribe the process. That is, the control of change is 'handed over' to actions within the program boundary (Chen et al., 2000a). When such actions are complex or additional system features (new inputs, outputs, etc) are required, it may be hard to make the appropriate changes and the process of revision and redesign will be costly. In the EM model the change of state is only achieved through automatic dependency updates or the direct action of an agent. Only when a pattern of state change has been experimented with and shows to operate reliabily, they can be delegated to an 'automated' agent action. This represents the progression from a subjective to an objective view of the system, starting from 1-agent modelling (corresponding to the modeller's view of system), then n-agent modelling (patterns of state change are identified and methods of communication are developed), and finally an 0-agent system is established whose meaning is independent of the modeller and the subject (Ness, 1997). This is why we choose the term 'empirical' to reflect the characteristic of our approach. Without such experimentation, the modelling process will become the reliable patterns of imagined state change as in conventional modelling.

The patterns of state change in observables are identified in EM through observation and experiment, which are also subject to revision within the subsequent observation. The observations of the EM model resemble the ones of the real-world situation. This is due to the interface of our model being for the purpose of the experimental interaction by the modeller in order to experience the development process, rather than just representation that the modeller views as a snapshot of the referent. Thus the

process of EM modelling involves incremental construction of artefact and modeller's knowledge which can be done by evolving the knowledge about the properties of the referent. The EM interface includes the EDEN input window in which new definitions can be entered, so the interface window (visualised by DoNaLD and Scout) can be used to animate the results or create an effect by introducing new definitions.

Conventionally the process of system development consists of three essential stages: requirements, design and implementation. However as discussed in chapter 2, this paradigm considers a serial rather than a concurrent approach to system development. All these stages are isolated from each other and normally failures will not appear until the implementation phase. The modelling process of EM provides an integrated environment for these stages in a development. That is, the EM modelling is a single 'process' rather than the traditional lifecycle and the same model can be used for understanding requirements of a system and its subsequent development. Thus in EM there is no requirements phase at the beginning of the development because such conceptual modelling is embodied in a script with a visualisation and can be experimented with by the modeller. Also, EM allows the on-line intervention for the modeller to experiment and establish reliable components which in some way resembles the testing task in conventional modelling. But such 'testing' in EM can be done earlier in advance of the whole application with the minimum of additional work and enable us to respond rapidly to the future evolution.

Finally, the interface of EM models serves the role of rapid prototyping which offers a 'mock-up' of a interface to the future system at an early stage (Chen et al., 2000a). This is similar to the use of an engineering prototype which is a physical object with characteristics similar to those of the actual system to be constructed. In other words, the process of EM modelling is the construction of a 'virtual prototype' (Beynon et al., 1996; Cartwright, 1998). The use-case driven approach OOSE (Jacobson et al., 1992) also has a similar user-interface prototype before developing the object model for the system (cf. p. 351), where the use case serves as an interface description to identify what information should be held and how it should be used. Typically in EM the design of interface is left until the purpose and requirements of the system have been clarified through the construction and exploration of the EM model.

4.4 Concluding Remarks

In this chapter we have described EM as a novel approach to modelling in a situated and open-ended manner. Based on the concepts of observable, dependency and agency, the process of EM modelling is the construction of an artefact, or a computer-based model, in a way similar to the way in which humans form a mental model of the real world subject. The primary focus of EM is the use of computer-based interactive situation models (ISMs) to represent the way in which systems are constructed in terms of these concepts. Thus far EM has been applied in artificial intelligence (Beynon, 1998), educational technology (Beynon, 1997), concurrent engineering (Adzhiev et al., 1994), software system development (Ness, 1997; Sun, 1999; Beynon et al., 1998; Beynon et al., 1999), geometric design (Beynon et al., 1996; Cartwright, 1998) and requirements engineering (Beynon and Russ, 1994; Sun et al., 1999; Sun, 1999). The ongoing research of applying EM includes decision support systems (Beynon et al., 2002), program comprehension (Beynon and Sun, 1998; Beynon et al., 1999) and BPR (the AMORE project).

To sum up, conventional modelling which formalises the development process and preconceives the possible states in the operational domain may be highly efficient at developing systems. But it may also restrict the flexibility of modellers in choosing alternatives and their openness to new concerns. In contrast, EM modelling aims to achieve *effectiveness* – measured by the overall outcomes of such a process rather than the speed or immediate results during the process – for the whole development. Further when broadly considering deploying computer systems into organisations, it is suggested by Winograd and Flores (1987) that the resulting design of systems, if concentrating on its efficiency, can erode the effectiveness of the organisation.

The potential application of EM to BPR is the current focus of this thesis and of work elsewhere (Chen et al., 2000a; Chen et al., 2000b). There are other methodologies for BPR, such as the process approach by Warboys et al. (1999) and the Object Advantage by Jacobson et al. (1995). Such methodologies have some characteristics similar to our EM approach, but there are some potential problems in

modelling business within them. As EM has been shown as a suitable approach to software system development and a tool for decision support and business application, we shall make a comparison between EM and other approaches for BPR in the next chapter to establish the link between system development and BPR. Based on this background, the framework of applying EM to BPR can be developed and will be detailed in the next few chapters.