Chapter 3

Empirical Modelling: A New Approach to Computer-Based Modelling

3.0 Overview

Chapter 2 presented an overview of the application of IT in the finance domain. This chapter focuses on a particular technology – Empirical Modelling Technology – developed at the department of Computer Science at Warwick University. Empirical Modelling technology provides a broad computational framework encompassing foundations for software system development, artificial intelligence, computer aided engineering design, business process modelling, and computer aided manufacturing. This puts EM technology in a position to potentially deliver solutions to problems in the finance domain.

Section 3.1 Introduces EM as a suite of key concepts, techniques, notations and tools. These are illustrated with reference to simple examples drawn from the finance domain. Section 3.2 highlights the distinctive qualities of model building in EM framed into: a) the focus on state as experienced; b) the use of artefacts for knowledge construction and c) the use of definitive scripts as a framework for distributed communication.

Section 3.3 concludes with motivating the use of EM to tackle technical and strategic demands for the wider agenda for computing.

3.1 Introduction to EM

«*Empirical Modelling (EM) is an approach to computer-based modelling that has been developed at the University of Warwick* [EM web site]. *It combines* ***agent-oriented modelling*** *with* ***state representation*** *based on* ***scripts of definitions*** *that capture the* ***dependencies*** *between* ***observables****. Unlike conventional modelling methods, its focus is upon using the computer as a* ***physical artefact*** *and modelling* ***instrument*** *to represent* ***speculative and provisional knowledge****. The central concepts behind EM are: definitive (definition-based) representations of state, and* ***agent-oriented analysis*** *and representation of* ***state-transitions****. In broad terms, changes of* ***state*** *within a system are interpreted with reference to a family of observables through which the interactions of* ***agents*** *in the system are mediated.»* [Bey99]

This section introduces Empirical Modelling as a set of principles, techniques, notations, and tools (cf. Table 3.1). It assumes some familiarity with programming paradigms and computer notations.

I. Empirical Modelling key concepts	**II. Empirical Modelling techniques**
• Observation / observable • Agency / agent • Dependency • State/ definitive representation of state • Definitive script • Agent oriented analysis • Representation of state transition	• Construe a situation • Construct an Interactive Situation Model (ISM) • Metaphorical representation through ISM
III. Empirical Modelling notations LSD account of observables Agent { state oracles handles derivates protocols }	**IV. Empirical Modelling tools** • EDEN interpreter • Distributed variant of EDEN

Table 3.1 Empirical Modelling Framework

3.1.1 Key concepts in EM

Empirical Modelling technology focuses on state representation. Empirical Modelling technology establishes principles that favour state representation over behaviour[1] automation at a preliminary stage of the software development process. Handling state is a major theme in computer programming [Bey98]. In EM, representing state in a comprehensible fashion is addressed in a definitive script that specifies the following information pertaining to state [Yun92]:

- *Observables as constituents of the state:* An observable is a characteristic of a subject to which an identity can be attributed. An observable in EM can be physical or abstract in nature. The clock is an example of a physical observable. The true value of a security in the financial market is an example of an abstract observable.
- *Dependencies between observables:* A dependency represents an empirically established relationship between observables. The attribute "empirically established" reflects the fact that a dependency is not merely a constraint upon observables, but reflects the impact of change in the value of one observable on other observables. Dependencies play a significant role in construing phenomena [Bey99].
- *Agents as instigators of state change:* An agent in EM is an instigator of change to observables and dependencies. An agency is an attributed responsibility for a state change to an observable. A literal dictionary definition [rhyme] of the term agent is "a substance that exerts some force or effect". The definition for the term agency is "the state of being in action or exerting power". These definitions indicate that an agent can be physical or abstract in nature, but it must be able to act or cause effects as granted agency by, and, for others or itself [Sun99]. For example, in the financial market, security price is an agent when it affects trading behaviour. In turn, trading behaviour acts as an agent when it affects security price. The agency of an entity cannot be specified with reference to its intrinsic features, since it is so widely open and undetermined [Sun99]. For instance, the exact location at which the price of a commodity is displayed on a board may influence how much that commodity is traded. It is very important to draw a clear distinction between the term agent used in the EM, and the term intelligent agent used in modern computer science. An intelligent agent is a computer system that is capable of flexible autonomous action in order to meet its design objectives. Flexibility means that the system must be responsive to change in the environment, proactive in goal directed

[1] A behaviour is a reliable repetitive pattern of action

behaviour, and able to interact with other artificial agents and humans [Jen97]. The difference between an agent in EM and an intelligent agent as defined by Jennings (1997), is in the degree of human intervention to play the agency role, and the openness of the agent action. The behaviour and functions of an autonomous intelligent agent are preconceived and well formulated in advance. An agent action in EM is situated[2] in nature and emerges from the modeller's private insight and perception of the real world.

- *Definitions to maintain state:* A definition in EM is similar in character to a formula in a spreadsheet (cf. Figure 3.1). Any change to the value of a dependee (a parameter of the built-in function used) will give rise to a re-evaluation of the dependent variable [Yun92].

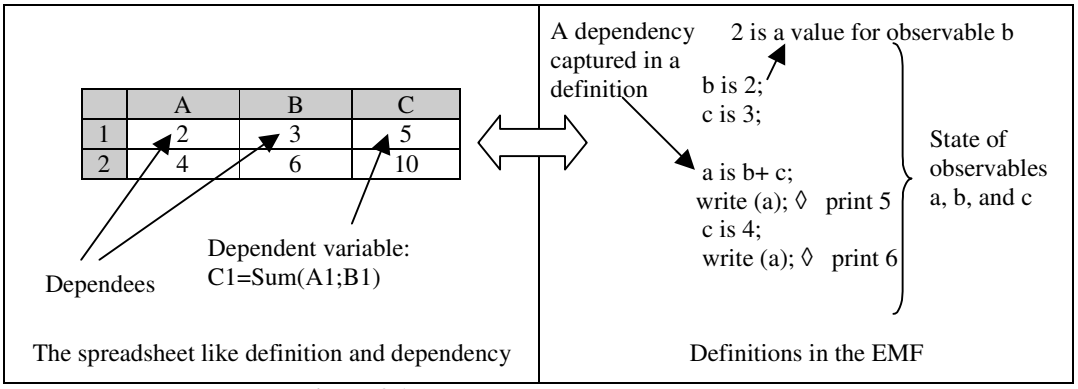

Figure 3.1 State and state representation in EM

State and definitive representation of state is illustrated by a simple two variables ordinary least square regression model developed using tkeden EM tool (refer to chapter 3 in the thesis web page on the Thesis CD). Definitions are used to maintain the state of the OLS estimates and residual errors represented by the observables beta_estimates, alpha_estimates, standard_error_alpha, and standard_error_beta. Part of the definitive script representing state in the OLS regression model is shown in the table below.

[2] An action is situated if it involves a conscious reference to context and choice of course of action. An action is not regarded as situated if it takes the form of a prescribed response or if it is an unconscious automatic response [Suc87].

```
%eden
Y is[10,15,36,18,19,20];  /* dependent variable taken as the annual
excess return on stock A*/
X is [12,56,78,98,90,67];  /* explanatory variable taken as the
annual excess market return  Rm*/

func sum_list_elements {
   para l;
   auto i, result;
   result=0;
   for (i=1;i<=l#;i++)
     result = result + l[i];
   return result;
}

func sum_square_list_elements {
   para l;
   auto i, result;
   result=0;
   for (i=1;i<=l#;i++)
     result = result + l[i]*l[i];
   return result;
}

func product_twolists_elements {
   para l,k;
   auto i, result;
   result=0;
   for (i=1;i<=l#;i++)
     result = result + l[i]*k[i];
   return result;
}


sumX is sum_list_elements(X);
sumY is sum_list_elements(Y);
sumsqrX is sum_square_list_elements(X);
sumsqrY is sum_square_list_elements(Y);
productXY is product_twolists_elements(X,Y);


/* computation of basic building blocks */
sumsqrx is sumsqrX -(1.0/(X#))*sumX*sumX;
sumsqry is sumsqrY -(1.0/(X#))*sumY*sumY;
sumxy is productXY -(1.0/(X#))*sumX*sumY;


/* computation of OLS estimates */
beta_estimate is sumxy/sumsqrx;
alpha_estimate is sumY/X# - beta_estimate*sumX/X#;

/* computation of the residual error */
sum_square_residual is sumsqry - beta_estimate*sumxy;
s_square is sum_square_residual/(X#-2);

/* computation of the error in the estimate for the regression
parameters */
standard_error_alpha is s_square*sumsqrX/(X#*sumsqrx);
standard_error_beta is s_square/sumsqrx;
```

Some basic functions operating on a list of elements, and returning a resulting list or number

A definition of observable as a value returned by a function

A definition of an observable as a formula

Table 3.2 Eden script for OLS Regression

The above model illustrates the use of a definitive script in EM and the similarity between dependencies in an Eden script and dependencies in a spreadsheet model.

Interaction with the script is open-ended. The dependent and independent variables can be changed and the list size can be increased by accepting the following re-definitions in the tkeden interpreter (cf. Figure 3.2).

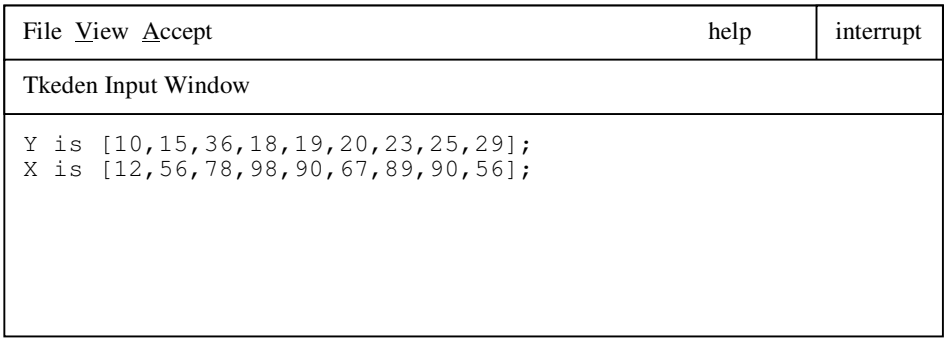

Figure 3.2 Introducing a definition / re-definition

Some advantages are gained when implementing OLS regression as a definitive script. First is the flexibility to change the values and size of the dependent and explanatory variables Y and X without the need to invoke any process to re-evaluate the estimation of the intercept and slope of the regression line. In an Excel spreadsheet, extending the range of the dependent and explanatory variables necessitate the use of the data analysis regression tool again with the new range of data. Compared to the use of a high level language like C, the Eden script is more flexible in the sense that a new definition of the dependent and explanatory variables Y and X can be accepted without having to include again the whole definitive script. The dependencies in the previously accepted script are maintained as long as they are not broken. This gives an added value over an implementation using a high level language, which requires running the program again with a new data set for X and Y.

The Eden script can be described as a radical generalisation of the spreadsheet concept in three respects. The first is in presentation because the dependencies are not only between variables in tabular format (cf. values in spreadsheet cells) but can be across any observable within the system. The second concerns the underlying data type because we can use abstract data types representing a far wider range than in a spreadsheet. The third is agency which allows dependencies to be handled by many human or automated agents concurrently [Roe00]. Establishing a link between the Eden script of the OLS regression and the graphical

representation of the regression line would demonstrate the generalisation of the spreadsheet concept beyond data in tabular format.

Depicting the efficient frontier[3] composed of portfolios with optimum risk-return trade-off using EM tools gives insight into the generalization of the spreadsheet dependency concept to apply to visual elements. A fuller description of the model related to portfolio diversification theory and the efficient frontier is found in the home page of chapter 3 in the thesis web page on the Thesis CD.

The definitive script establishes dependencies between observables that typically have some form of visualisation (point, text message, window, 2D visual metaphor) attached to them. The definitive script shapes the semantics of the visual elements attached to it.

The Eden notation is used to develop a script establishing dependencies between observables. The Donald notation is used for graph drawing, and the Scout notation is used for screen display. The script applies for a portfolio of two assets. Extension of this script to account for more assets would need additional definitions for the calculation of variance and covariance of returns.

The two classes of assets (their expected returns and probability of occurrence of these returns) are represented in the abstract data type list structure in the Eden notation:

```
/* r=[[return , probability of its occurrence in state 1 of the economy],..];*/
R1=[[20,0.2],[5,0.6],[-10,0.2]];
R2=[[50,0.2],[15,0.6],[-20,0.2]];
Rf= 12; /*the risk free rate of return*/
```

Functions for the mean[4], variance[5], covariance[6] and correlation coefficient of returns are developed. The variance matrix and its inverse are formed. The efficient frontier[7] is a scatter graph relating portfolio returns (y axis) and portfolio variances (x-axis).

[3] This relates to portfolio theory first developed by Harry Markowitz in 1952. The thinking behind the explanation of the risk-reducing effect of spreading investment across a range of assets is that in a portfolio unexpected bad news concerning one company will be compensated for to some extent by unexpected good news about another. Markowitz provided us with the tools for identifying portfolios that give the highest return for a particular level of risk. Investors can select the optimum risk-return trade-off for themselves depending on the extent of personal risk aversion [BKM96, Arn98, EG95, CLM97].

[4] $\bar{r} = \sum_{i=1}^{n} r_i p_i$ where $\bar{r}$ is the mean return; r_i is the return of the security if event i occurs;

The Eden model developed allows the exploration of different return/variance combinations. Additional script can be easily added to visualise the security market line[8], and capital market line[9]. All observables are linked to each other by dependency relationships. New values given to observables will automatically update all the dependent observables as well as the graph.

```
invsigma11 is sigma22/(sigma11*sigma22 - sigma12*sigma21);
invsigma12 is sigma12/(sigma12*sigma21 - sigma11*sigma22);
invsigma21 is sigma21/(sigma12*sigma21 - sigma11*sigma22);
invsigma22 is sigma11/(sigma11*sigma22 - sigma12*sigma21);
inverse_sigma_matrix is
[[invsigma11,invsigma12],[invsigma21,invsigma22]];
```

Any observable value, can be queried at any time. A visualisation is attached to the two observables portfolio returns and portfolio variances that are linked by the dependency relationship defined by the efficient frontier equation. The definitive script establishing a dependency between Eden observable and Donald variables that define the shape and the semantic of visual elements is presented in the table below.

```
%eden
B is invsig1[1]*mu1+invsig1[2]*mu2;
A is invsig1[1]+invsig1[2];
C is mu1*invsigr[1]+mu2*invsigr[2];;
D is A*C - B*B;
func efficient
```

and p_i the probability of occurrence of event i.

[5] $\sigma = \sqrt{\sum_{i=1}^{n} (r_i - \bar{r})^2 p_i}$ where σ is the variance of the security ; $\bar{r}$ is the mean return of the security; r_i is the return of the security if event i occurs; and p_i the probability of occurrence of event i.

[6] $\text{cov}(R_A, R_B) = \sum_{i=1}^{n} (RA_i - \overline{RA})(RB_i - \overline{RB}) * p_i$

where cov(R_A, R_B) is the covariance of the two securities A and B ; RA_i and RB_i are the returns of securities A and B if condition i occurs; $\overline{RA}$ and $\overline{RB}$ are the mean returns of securities A and B; p_i is the probability of occurrence of condition i

[7] $\sigma_p^2 = \frac{(\mu_p - r_f)^2}{Ar_f^2 - 2Br_f + C}$ where σ_p is portfolio variance; r_f is the risk free rate of return; μ_p is the portfolio return; A, B, and C are calculated from a unique combination of the risk-free asset and the "tangency portfolio" which maximises the expected utility of the investor's end -of-period wealth.

[8] A linear line showing the relationship between systematic risk and expected rates of return for individual assets (securities). According to the capital asset pricing model, the return above the risk free rate of return or a risky asset is equal to the risk premium for the market portfolio multiplied by the beta coefficient.

[9] The set of risk-return combinations available by combining the market portfolio with risk free borrowing or lending.

```
{
para mu;
auto va;
val=sqrt(((A*mu*mu)-2*B*mu+C)/D);
return val;}
proc constructefficient
{auto val;
MUy=[0,0.05,0.1,0.15,0.2,0.25];
Sigmax=[];
for (i=1;i<=MUy#;i++)
{
  val=efficient(MUy[i]);
  Sigmax=Sigmax//[val];
  }}
constructefficient();
  P is Sigmax;
  Std is Muy;

%donald
viewport drawgraph
int n,xsc,xsh,ysc,ysh
n=5
xsc=100
ysc=50
xsh=50
ysh=50
graph main, xaxis, yaxis
within main {
  x<I>=getx!( <i>)

  f<I> = gety!(<i>)
  nSegment = ~/n
  node = [circle:circle({x<i> *~/xsc+~/xsh, f<i> *~/ysc+~/ysh}, 10)]
  segment = [line:[{x<i-1>*~/xsc+~/xsh, f<i-
1>*~/ysc+~/ysh},{x<i>*~/xsc+ ~/xsh, f<i>*~/ysc+~/ysh}]]
           }

%eden
proc graphChange : std,P

{maxx=findmaxx();
 minx=findminx();
 maxy=findmaxy();
 miny=findminy();
 _n is (P# );
 scalexy();
 }

 graphChange();
```

Table 3.3 Eden, Donald, and Scout scripts for the CAPM

The following figure shows the screen display of the efficient frontier linked to observables in the model.

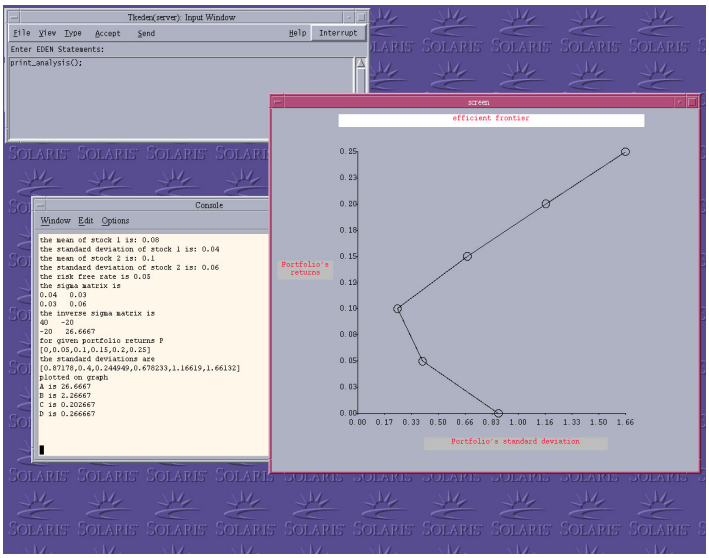

Figure 3.3 Use of Eden, Donald and Scout to explore the efficient frontier

- *Agent oriented analysis:* Agent oriented[10] analysis in EM relates the interaction between agents - in the first instance - to basic perception of observables. More sophisticated issues of knowledge representation can follow from this preliminary agent oriented analysis after the identification of a reliable and persistent mode of interaction between agents. For example, in a financial market context, a trader would resort to an agent-oriented analysis to gain basic knowledge of the behaviour and interaction in the market. Once this basic knowledge is established, more sophisticated mathematical modelling would be more appropriate. This perspective on financial modelling is consistent with Gooding's (1990) point of view on the importance of observation and experimentation in testing the truth and validity of new theories and technologies. The concept of agent action in EM is complementary to the notion of indivisible change of system state as expressed in definitive scripts [BJ94]. State in EM is represented by means of a system of definitions, each of which defines the value of a variable either explicitly, or implicitly in terms of other variables and constants. Transitions from state to state are performed by redefining variables [BNRSYY89].

[10] Note that the agent oriented approach in the EMF differs from agent orientation as referenced in AI research such as [Sho90].

3.1.2 EM techniques

The concepts of observables, agency, dependency, definitive representation of state and agent oriented analysis in EM support experiential knowledge construction of an application domain by developing a computer artefact / a cognitive artefact based on construing a situation and constructing an associated Interactive Situation Model (ISM).

A literal dictionary definition [rhyme] of the term construal is "an interpretation of the meaning of something". In EM, a construal is represented metaphorically via a physical artefact, typically computer-based, and has a number of key features [Bey99]:

- *It is empirically established (it is informed by past experience and is subject to modification in the light of future experience);*
- *It is experimentally mediated;*
- *The choice of agents is pragmatic (what is deemed to be an agent may be shaped by the context for our investigation of the system); It only accounts for changes of state in the system to a limited degree (the future states of the system are not circumscribed).*

This interpretation of construal has a similar meaning to that introduced by Gooding (1990):

> *"Construals are a means of interpreting unfamiliar experience and communicating one's trial interpretations. Construals are practical, situational and often concrete. They belong to the pre-verbal context of ostensive practices".*

Beynon (1999) identifies an important difference between construing a system in the EM framework and in the AI framework. In the AI framework, a system is construed as "acting as if it were inferring", a mathematical structure of objects is adopted, and preconceived functions for a system to achieve its purposes are presumed. A construal in EM is represented metaphorically via a computer-based physical artefact constructed based on previous experience and is subject to exceptional behaviour for which there is no pre-conceived explanation. In the Empirical Modelling framework a system is construed, "as if it were composed of a family of agents, responding to observables, and exercising privileges to change their values in the context of a set of dependencies between observables"

A literal dictionary definition [rhyme] of the term situation is "a condition or position in which you find yourself". In EM, activities are considered as "situated". The term "situated activity" introduced in [Sun99] refers to a coherent sequence of situated actions that is

constructed by the interaction between a human agent and its environment[11]. In our daily life, we envisage many situated activities. Examples of these are illustrated by considering different scenarios drawn from a financial context:

- buying a portfolio of shares
- visualising a financial data set
- joining a group conversation on a debatable issue in finance such as market efficiency.

In the first scenario, the investor might conduct a financial market analysis to support his/her stock selection, consult a broker and delegate to him/her the task of portfolio construction, or make a random selection of shares. In the second scenario, the investor might be satisfied with a rough paper sketch, resort to graph drawing utilities in a spreadsheet application, use a special purpose graphic package, or write code in a high level programming language to visualise the data set. The third scenario involves listening and replying to the speaker.

An action is situated if it involves a conscious reference to a context and a choice of a course of action. A situated activity is different in character from an activity specified by a formal algorithm. A situated activity is difficult to prescribe in advance. This difficulty is revealed by attempting to answer, in advance and with full certainty, the question of how this will be done in each of the above considered scenarios.

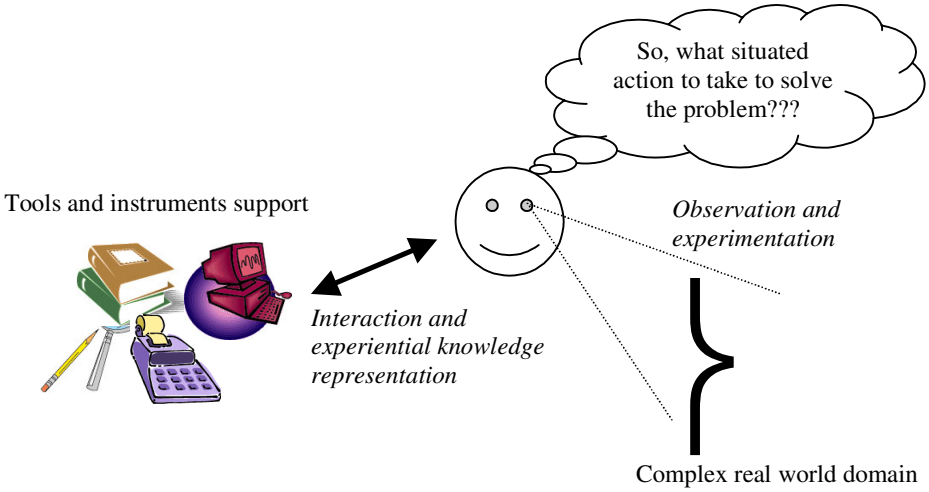

Figure 3.4 Solving problems in the real world domain

Solving problems in the real world domain is a situated activity rather than a formal activity, especially when problems are first encountered. Suchman (1987) argues that most plans (algorithms, strict laws, formal methods) used by human agents serve as a resource rather than

[11] Refers to the external surroundings of an individual.

a source of control in everyday life (cf. Figure 3.4). This argument is also supported by the fact that a solution to a real world problem is context dependent (i.e. it cannot be detached from the real world and abstracted in a formal algorithm), and human centred (i.e. the human role is central, is not pre-conceived at an initial stage, and is difficult to capture in formal logic or rules).

However, a situated activity is error prone because it relies upon human discretion, as compared to a formalised process derived from an engineering discipline. Hale (1998), Norman (1983), and Radford et al (1974), recognise the fact that the human being is inevitably error prone and forgetful, learns slowly from experience, and can be seriously distracted by the external environment. These human factors highly influence the structure of the situated activity. The development of tools and instruments aims at supporting the human activity in reducing the impact of human weaknesses. However, in the case of computer-based tools, formalized interaction and context-independent algorithms limit the effectiveness of these tools in empowering human strengths.

The EM technique of construing a situation entails construing a system in a situated way. That is a system admits different construals, each formed based on a situated judgement. An Interactive Situation Model (ISM) is developed to explore different construals. The ISM is a computer-based environment constructed through a situated modelling activity. Unlike a closed-world computer model with a fixed interface, an ISM is always open to elaboration and unconstrained exploratory interaction [Bey94]. States within the ISM metaphorically represent pertinent situations from the application domain, and possible transitions between states are explicitly constructed so as to be consistent with the developer's construal of a system in terms of agents, observables, and dependencies [BCSW99].

Interactive Situation Models were first introduced and used in EM to assist in the software development process, and in particular in the requirement engineering phase of this process. The use of an ISM was further extended to support the development of reactive systems, and for providing computer-based support for diverse activities in the real world domain including: business process modelling, product design, learning, decision support, and interpersonal communication.

The use of an ISM is illustrated in this chapter, and in chapter 5, with reference to the story of a retail trade in NYSE extracted from [Har98]. The purpose of the ISM developed for this story is the exploration of the trading mechanisms in the financial market and the design and organization of financial markets. The trading story can be referred to in the home page of chapter 3 in the thesis web page on the Thesis CD.

Constructing an ISM for a NYSE retail trade is a way of modelling an external observer's explanation of the retail trade process (RTP). In its most naïve form, such an explanation explicitly relates the actions of agents to the stages of the trading protocol. This simply involves identifying the actions for which each agent is responsible, and identifying the preconditions under which each action is performed. The major roles in a retail trade are played by the investor and the broker. The *investor* requests information on a particular stock from the broker, puts a trading order, confirms his order, pays for his transaction, and acquires or releases share ownership following the execution of his order. The *broker* requests quotes from the quote information system, returns this information to the investor, enters any received order in the order entry system, reviews the order details prior to its release in the order entry system, reports the trade execution to the investor, receives payment including charge fees, and mediates the exchange of share ownership. Each of these actions on the part of investor and broker is performed at a specific stage in the RTP. The following figure is a screen snapshot of an ISM built using the EDEN interpreter.

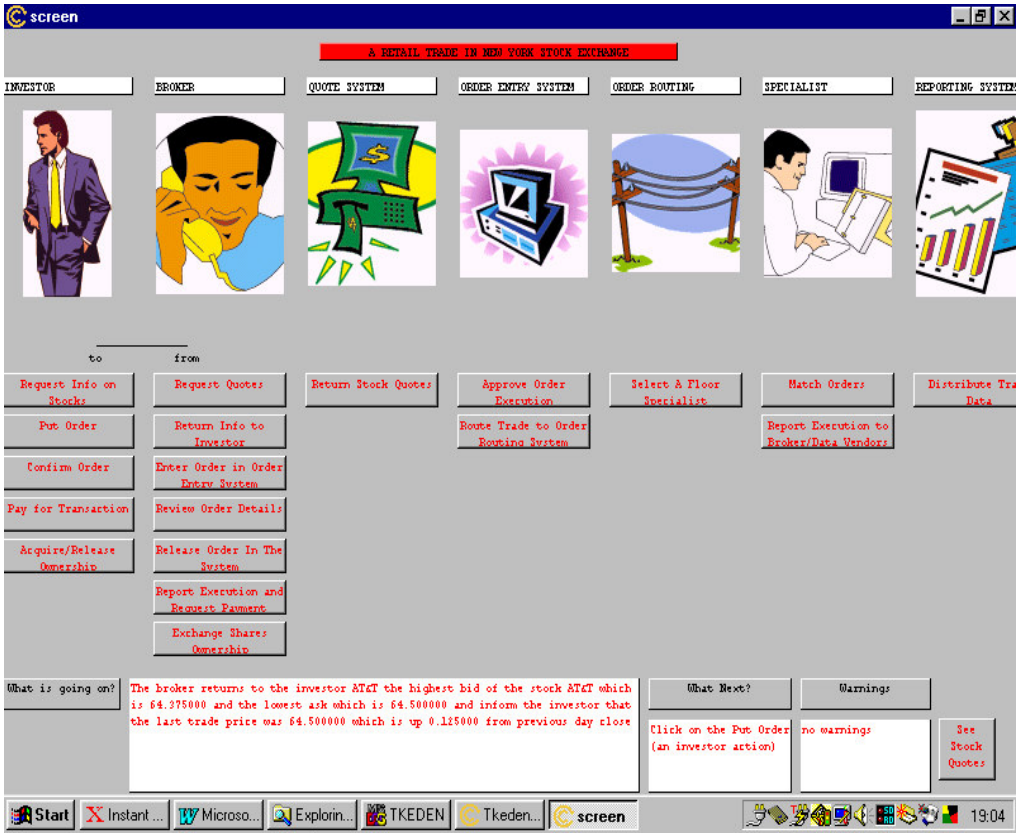

Figure 3.5 A snapshot of the ISM for a retail trade in NYSE

3.1.3 EM notations

In Empirical Modelling, all state-changing activity is attributed to agents. An agent can be a human actor, a state-changing process or component. The role of an agent can be played by a modeller, or by the computer. In the process of explanatory analysis of a situation, many aspects of the agency and dependency that are being identified will be implicit in the interaction between the modellers and the computer artefact. In understanding the behaviour of a system, it is also typically important to identify explicit protocols and stimulus-response patterns that are characteristic of agent interaction [Bey99, BM00].

A special-purpose notation - the LSD notation - has been introduced to describe such agency. An *LSD account* is a classification of observables from the perspective of an observer, detailing where appropriate:

- the observables whose values can act as stimuli for an agent (its *oracles*);
- those which can be redefined by the agent in its responses (its *handles*);
- those observables whose existence is intrinsically associated with the agent (its *states*);
- those indivisible relationships between observables that are characteristic of the interface between the agent and its environment (its *derivates*).
- what privileges an agent has for state-changing action (its *protocol*).

The use of an LSD account is illustrated with reference to the model of a retail trade in New York Stock Exchange considered earlier.

The roles of the various agents in the NYSE have to be understood in terms of the relevant observables. Some of these observables (such as the current status of a BUY/SELL order) are particular to the retail trade situation, but the actions of agents also relate to observables generic to the online trading context.

In the online trading context, the social network comprises investors, brokers, dealers, arbitrageurs, and boards of trade. The trading marketplace may be a physical trading floor or an electronic system. In the retail trade situation, the relevant ***agents*** in the model are identified as: the *investor*, the *broker*, *the physical stock exchange*, *the company stock the quote information system*, the *order entry system*, the *order routing system*, the *floor specialist*, and the *information reporting system*. The relevant ***observables*** for the participating agents comprise:

- *Order information*, including: investor name, ID, BUY/SELL order, share name and symbol, quantity of shares, type of order (such as market, stop loss, limit order, etc.), price (if needed), expiry date of the order, the date and time of the order.
- *Stock quotes*, including: stock symbol, bidder, BID/ASK, price, size, time and date.

- *Stock information*, including: stock symbol, stock name, last trade price, change from previous day close, time last traded, place last traded, highest day price, lowest day price, day volume.
- *Order indications from dealers and brokers*, including: the stock name, the name of the broker/dealer, the time, and the date.

A possible account of the broker's response to an information request might be:

1. check status of request information action by investor;
2. get investor information request;
3. direct information request to quote information system;

update current RTP status;

The current stage reached in the RTP is interpreted as an observable for the participating agents. Each agent action is formulated in terms of re-definitions of observables. For instance, in the initial stages of the RTP, the broker requests quotes from the quote information system when an investor has requested information on a particular stock.

The following table presents the LSD notation used to describe the agency and dependency in the account of the story of a retail trade in New York Stock Exchange.

The LSD template for describing the broker agency in the account of a retail trade in New York Stock Exchange

agent broker	{
state	info_requested, quotes_info_requested, Commission rate, bid and ask price (if the broker acts as a market maker or dealer), trade history, personal account (profit account = cumulated commission revenue + revenue from spread (in case broker is dealer)
oracle	stage_in_retail_trade info_requested Investor's orders, price change, order status
handle	order status, bid and ask prices of a stock, portfolio holding of investor, commission rate
derivate	stage_in_retail_trade = F(info_requested, …) ….
protocol	(stage_in_retail_trade = init_trade) and (info_requested) ◊ quotes_info_requested=1 (order_issued=1) ◊ validate_order( ) (order_approved=1) ◊ order_directedtorouting system=1 (order_directedtoroutingsystem=1) ◊ order_directtofloorspecialist=1
}	

Table 3.4 LSD account for the broker agent in the story of a retail trade in NYSE

3.1.4 EM Tools

The principal EM software tool developed so far is tkeden. Tkeden can interpret three types of notation: Eden, Scout, and Donald.

The modeller viewpoint is represented by a script of definitions (a *definitive script – Eden script*) resembling the system of definitions used to connect the cells of a spreadsheet. The variables on the LHS of such definitions are intended to represent observables associated with the external situation. There is typically some form of visualisation attached to them, so that for example a variable can denote a point, a text message or a window displayed on the computer screen. Scout and Donald notations are used to attach a 2D visualization to Eden script.

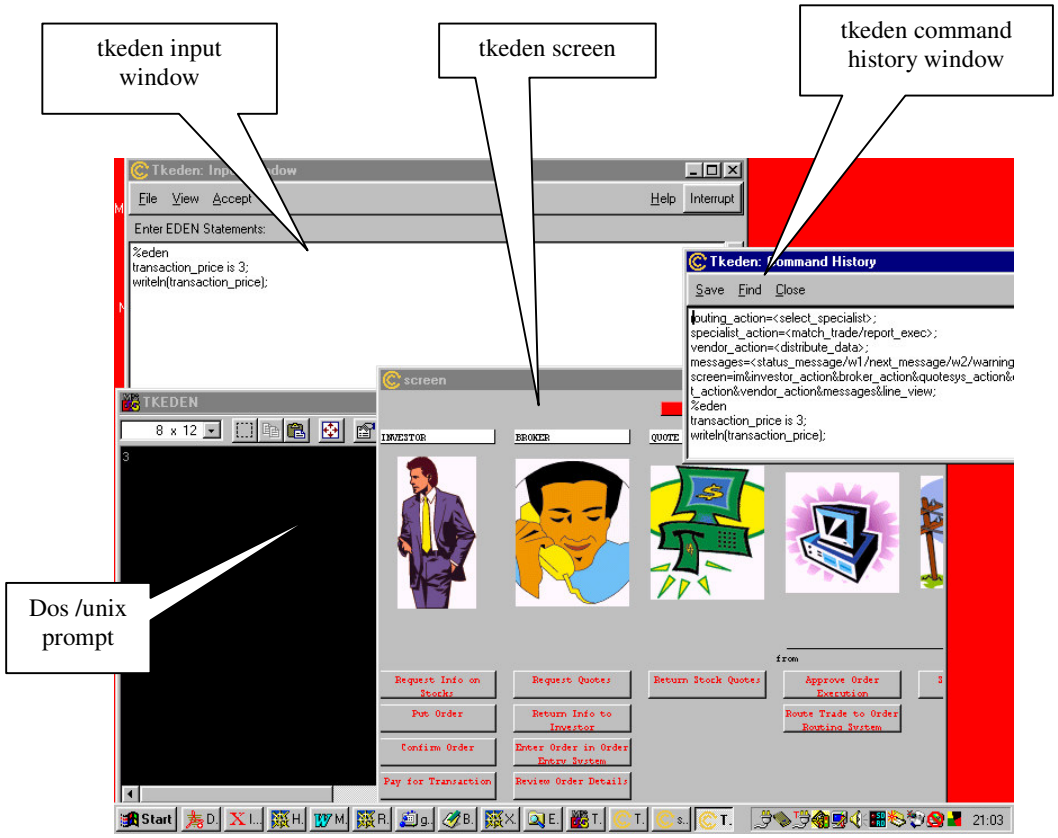

Figure 3.6 The tkeden interpreter

The tkeden interpreter is composed of the following components:

- An input window, where Eden, Scout, and Donald notations are edited and accepted.

- A command history window, that stores all commands accepted by the tkeden input window since the start of the current modelling session.
- A screen, where Scout and Donald notations are visualized using 2D geometric metaphors including buttons, text input, lines, and circles.
- An output window for displaying the definitions and values of variables.

A distributed version of tkeden has also been developed: dtkeden. It is implemented on a client-server architecture, in which the viewpoints of individual modellers are represented by independent definitive scripts executed on different client workstations. State changes are communicated between clients by sending redefinitions across the network via the server. Communication strategies can be specified via the server to suit different purposes. The server can play a role in negotiation between clients, resolving conflicts or dictating the pattern of interaction and privileges of modellers.

Distributed Empirical Modelling is illustrated with reference to the case study of a distributed Stock Market Game. The game was originally developed by the author, and tailored to the style of trading in London Stock Exchange by Ajul Shah. This development of the game is different in spirit from a game theoretic approach[12]. Whereas game theory is more sophisticated in analysing multi-person decision making as described in [Gib92], it gives limited scope for experiential knowledge and personal insight.

Observation and experiential knowledge are used to develop the game. The identification of players in the game is subjective and reflects personal insight and basic understanding of trading behaviour. The development of this game illustrates a basic application of Distributed Empirical Modelling (DEM) technology to modelling financial markets. A very simple model of electronic trading emerged with little support for the understanding of some basic aspects of stock trading. The initial model does not take account of trading rules and regulations in any market. A distributed version of tkeden, dtkeden, is used to implement the model on four workstations.

[12] Game theory is the study of multi-person decision problems. Such problems arise frequently in economics. At the micro level, models of trading processes (such as bargaining and auction moodels) involve game theory. Labor and financial economics include game-theoretic models of the behaviour of a firm in its input market. There are also multi person problem in a firm. Games are classified into classes: static games of complete information, dynamic games of complete information, static games of incomplete information and dynamic games of incomplete information. Four notions of equilibrium are in these games: Nash equilibrium, subgame-perfect Nash equilibrium, Bayesian Nash equilibrium, and perfect Bayesian equilibrium [Gib92].

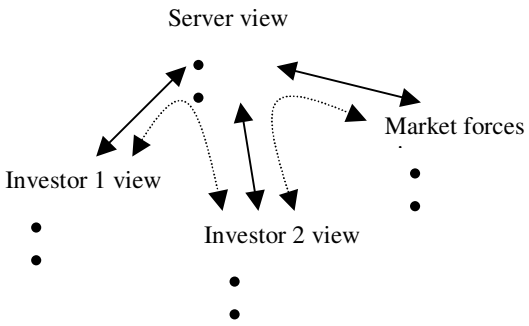

Figure 3.7 The star configuration of the distributed game

A star-type configuration for network communication (cf. Figure 3.7) is adopted to link a server to three clients representing two investors and the action of market forces. Each investor can monitor the prices of the securities within their view. They can trade (buy/sell) these securities. Some business rules in trading are to be respected (an investor cannot sell a share that he does not own; an investor cannot buy a number of shares of a particular firm exceeding the number of shares issued by the firm). The **`market forces`** agent simulates all the events affecting the prices of traded shares and the decision made by listed firms to issue an additional number of shares. The **`market forces`** agent can change the price of the shares up or down and it can increase the number of issued shares by a given firm. Each investor is supposed to construct his/her own portfolio by buying and selling shares from the market. An intelligent investor would buy shares if shares are under-priced and would sell shares if they are over-priced. The portfolio balance of an investor is the market value of all the shares within his/her ownership. The financial position of investors is the sum of their cash holding and the market value of their portfolio, which is subject to change under the action of the **`market forces`** agent. The **`market forces`** agent can increase the number of issued shares by a firm thus giving the investor an opportunity to buy a larger number of shares issued by a particular firm. A sample of the client-server communication is shown in Figure 3.8 below. It illustrates distributed communication through definitive scripts. The **`market forces`** agent sends a new value for the price of share 1 to the client **`investorI1`**. This will change the value of **investorI1_balance** that depends on the value of the price of share 1 via a definition.

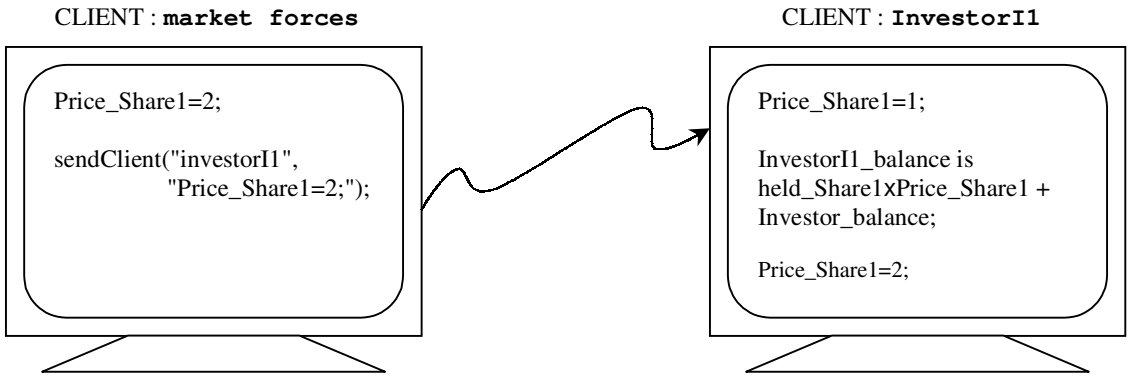

Figure 3.8 Distributed communication of definitive scripts

Where interaction is concerned, dtkeden allows transmission of re-definitions between scripts located at the server or the clients according to the star-type configuration for network communication. The semantics of such a re-definition is wide open: it encompasses actions that assign different values to observables or create dependencies between these observables. New observables can be introduced, such as might represent external factors affecting market behaviour. The Scout interface handles text-based interactions, for example, for buying and selling shares and for displaying the winner/loser. The most primitive but powerful mode of interaction with an EDEN interpreter is through entering definitions directly into an input window. In principle, all market participants can exploit input of this nature. The investors can send messages to each other via the server, and the server can broadcast general news to all market participants. It is possible to specify which observables values are **handles** and **oracles** in LSD terms, and accordingly then can be changed and/or inspected by a market participant [BM00, Sun99].

The quality of communication in dtkeden stems from the fact that the representation of state is definitive, transitions are effected by transmitting re-definitions and each interpreter actively maintains and monitors the current state of relevant observables. All these features reflect the way in which agents are construed to act and interact in the real-world. They are also relevant to architectures for agency. Where present computing platforms are concerned, the need to distribute EDEN interpreters may be regarded as imposing a significant computational demand on each client. There are ways in which this issue could be addressed – in particular, by localising and customising dependency maintenance to balance the resources available for reconstruction of state and the bandwidth for transmission of state information.

A comparison with other programming technologies that might be applied to model virtual trading is helpful. Current web technology is highly document-centric. To transmit non-textual data requires techniques such as pre-process-and-publish that are far from delivering the direct influence over remote state that the transmission of a re-definition effects. The scope for interactive agency in a web network is inhibited by the standard net protocols: the state of a webpage is updated only when the viewer of the webpage initiates a request. With conventional software development methods, a Java implementation of virtual trading would be targeted at a specific preconceived requirement for market participant interaction. In this connection, there is a trade-off between the narrowness of the requirement and the quality and efficiency of the solution.

A brief account of how the model is constructed illustrates how its functionality remains open-ended. As in all EM models, there are many different ways in which the constituent definitive scripts, functions and actions can be organised into clusters. These can correspond to conceptual layers in the model, to submodels suitable for re-use, or to partitions into observables associated with specific sub-objects for instance.

The above simple distributed model simulates the behaviour of an uninformed[13] investor in an inefficient market. The market is inefficient because current prices do not necessarily reflect publicly available information. The investor is prone to high loss as well as high gain on a speculative basis. However, the price reporting and trade execution process is highly efficient, as current prices are directly appearing on the screens of the investor, and the transaction is executed automatically without delay. This relates to research in finance in the area of market microstructures. The use of EM technology to model financial market microstructure will be considered more deeply in chapter 6.

The model so far developed is unrealistically simple. In the real world, the delay in transmitting actual price changes to investors, the delay in trade clearing and settlement, and the lack of market transparency, all affect the trade process. The model can be extended to account for all these factors. Moreover, decision support for the investor and intelligent analysis can be incorporated in a more advanced model.

Shah (2000) introduced some improvement to the model, bringing it closer to real trading in a specific exchange and extending its capacity (more investors can connect to the server, more stocks are considered, a larger source of market information is provided, and additional visualisation is introduced for the investor's portfolio and financial indicators). The model is a

[13] An uninformed investor is an investor who does not perform any analysis before executing any trading transaction.

prototype for a multi-player game in which each player has the ability to analyse information and then buy and sell shares. The mechanism of the trade is more realistic. Trading rules for order matching are followed. A model simulating SETS[14] is also developed separately.

3.2 Distinctive qualities of Model Building in EM

The principles, techniques, notations and tools of EM exhibit the following distinctive characteristics:

a) the focus on state as experienced;

b) the maintenance of a semantic relationship between an application domain and a computer-based artefact;

c) the use of an artefact for knowledge construction;

d) the use of definitive scripts to support collaborative distributed modeling.

a) Empirical Modelling focuses on state as experienced rather than state as abstracted

State in EM vs state in mathematical/ conventional model

There are key differences between the representation of state in EM and in abstract mathematical / conventional models. These differences are attributed to several factors and considerations related to: i. entities in the model; ii. relationships between entities in the model; iii. human agency; and iv. the scope for distinguishing different aspects of state

i. The primitive entities in EM are observables that have counterparts in the real world. Entities in mathematical / conventional models are variables that are abstract representations of observables in the real world. Once identified, they constitute the basic elements of the abstract representation of a real world domain.

   In EM, observables can be added to the computer-based model to reflect our growing ***experiential*** knowledge of new observables in the real world. There is no preconceived description of, nor limit on, the number of observables in an EM model.

[14] SETS (Stock Exchange Electronic Trading Service) is the LSE's fully electronic order book trading mechanism. The order book is the central price formation and trading mechanism for the securities in the FTSE-100 index, reserve securities and others. There are no market maker quotes for these securities. The order book allows participants to submit orders displaying their willingness to buy or sell share at specific prices, or to execute against displayed orders. Execution occurs when a buy and a sell order match. Orders are submitted by stockbrokers either for clients or for themselves.

In mathematical / conventional models, abstraction involves focusing on particular aspects of the real world domain and on simplifying reality for ease of understanding and representation. This limits the number of variables in mathematical / conventional models.

Observables in EM can be agents (instigator of state change), oracles (seen by other agents) or handles (manipulated by other agents). No particular characteristics can be attributed to the variables in a mathematical model apart from having a particular value determined by the specific function that they are introduced to serve.

ii. In EM, relationships between entities (observables) are established through definitions to reflect particular relationships between observables in the real world. These do not take the form of absolute invariant relationships between values of variables but of dependencies that express the modeller's current provisional expectations about how changes to some observables indivisibly affect the value of other observables. In particular, relationships between observables in the real world are not permanent in time. This reflects two factors: first, our knowledge of the relationship between observables in the real world is not perfectly exact; and second, observables in the real world may undergo change, and hence their relationship might change as well. As such, relationships between observables in EM can be altered by new definitions or re-definitions to reflect change encountered in the real world. In a mathematical / conventional model, relationships between variables are pre-conceived and formally established. There is no point in adding new relationships or altering existing ones as the determination of these relationships is strictly defined by the abstract representation of the real world.

iii. Human agency is central in EM. An EM model assumes the existence of a super-agent (the modeller – a human) who observes and interacts with the real world and the computer-based model. This super-agent sees the real world from a personal subjective view and constructs the computer-based model according to his/her own view. The growing experiential knowledge of the human modeller of the real world domain enriches the model with observables and relationships between observables and the identification of agents and agent actions. Observables, relationships between observables, and agents in the model are determined subjectively by the human modeller and are subject to change with growing knowledge and evolving modes of observation. In a mathematical / conventional model, variables and relationships between variables are objectively determined to reflect an abstract representation of

the real world that is based upon preconceived modes of observation and interpretation. Human intervention is permissible to change the values of certain variables following prescribed actions and through appropriate interfaces.

iv. State in EM is subjective (reflecting the modeller's view of the external world), situated (reflecting the actual situation in the real world) and context dependent (strongly related to the real world context to which it refers). As such, state in EM reflects state as experienced in the real world. State in a mathematical conventional model is an abstract state detached from its real world context. It hardly permits experiential knowledge construction.

Illustration The interaction with an EM model is open ended and resembles our interaction with the real world. This is illustrated with reference to an example[15] of the determination of the price of a security in the financial market, where P refers to the price of the security, DD to its demand, and SS to its supply. In the context of this example, an EM model reflects how our understanding of the relationship between P, DD, and SS can evolve. The modeller might regard DD and SS as things that are observed and that can be used in establishing P. He / she might think of DD and SS as agents that affect P according to some protocols and might identify a relationship between P, DD, and SS. Such a relationship can be re-defined to reflect evolving knowledge of P as new observables, such as *economic conditions,* are considered in the model.

In comparing the EM model with a mathematical / conventional model for this example, the differences are clearly drawn:

- Entities in the EM model are the observables P, DD, SS that are determined subjectively by a human modeller who is monitoring the state of the financial market and interacting with the computer-based model. Whereas entities in the mathematical / conventional model are the variables P, DD, SS that are objectively determined in an abstract representation of the real world (financial market).
- Relationships between entities in the EM model are empirically established by the modeller following his / her experiential knowledge of the state of the financial market. Such relationships may take the form of definitions such as: **`P is f (DD, SS)`**. Relationships between variables in the mathematical / conventional model are determined by the abstract representation of the real world and take the form of an expression that can be evaluated according to the values of its variables : **`P= f (DD, SS).`**

[15] The considered example is hypothetical and does not derive from a particular theoretical foundation in finance.

- Observables in the EM model are classified as agents, handles, and / or oracles. DD and SS may be considered as agents affecting the state of P, and P as a handle and oracle for DD and SS. An LSD description can be used to capture the description of agency and dependency in the model. The LSD description for DD may take the form of:

```
Agent DD
{
   State  Value_of_DD
   Oracle P
   Handle P
}
```

  Variables in the conventional / mathematical model can only take on the values determined by the relationships in the abstract mathematical model.

- The state of the EM model is situated, context dependent, and subjectively determined by the modeller. Growing experiential knowledge of the modeller alters the state of the model. The modeller might conceive a new observable (economic condition) affecting the state of the model and introduce a re-definition in the model such as:

  P is f (DD, SS, economic condition)

  State in the mathematical / conventional model is determined by its behaviour (a repetitive abstract state).

b) Empirical Modelling aims at maintaining a semantic relationship between a computer-based artefact and an application domain that reflects the modeller's construal

Motivation

Empirical Modelling addresses the problem of the separation between experiences of the real world and of the computer-based model. Beynon (et al, 2000) argues that such a separation may be less of a problem in scientific or engineering applications where theories and abstract entities can be successfully applied to a certain extent. But in social and business domains such a separation leads to major problems. This stems from the difficulty of applying contextual information appropriate to different situations and the difficulty of end-users[16] to modify the models.

The semantic relationship in EM

The focus on state as experienced and the continuous human engagement in the modelling activity in EM implies a semantic relationship between the computer-based model and its corresponding real world referent. Such a semantic relationship can potentially support semi-automated activities where human input and agency is paramount.

[16] End-users refers to managers in a business context.

In EM, interaction with the computer-based model can be directly compared with real world experience. This interactive experience can be mediated by metaphors[17] or by a virtual reality style of interface[18]. The main objective of the computer-based model is to cultivate the understanding of the modeller of his experience in the real world. Experiential knowledge about the state in the application domain can be represented by experiential knowledge about the state in a virtual environment. A semantic relationship would then be established between the two experiences [BRR00-1]. Experiential knowledge about state in the application domain is subjective and needs a medium to expose it publicly, however, experiential knowledge about state in a virtual computer-based environment can be to some extent recorded by information processing mechanisms. This emphasises the use of the computer as an instrument for experimentation.

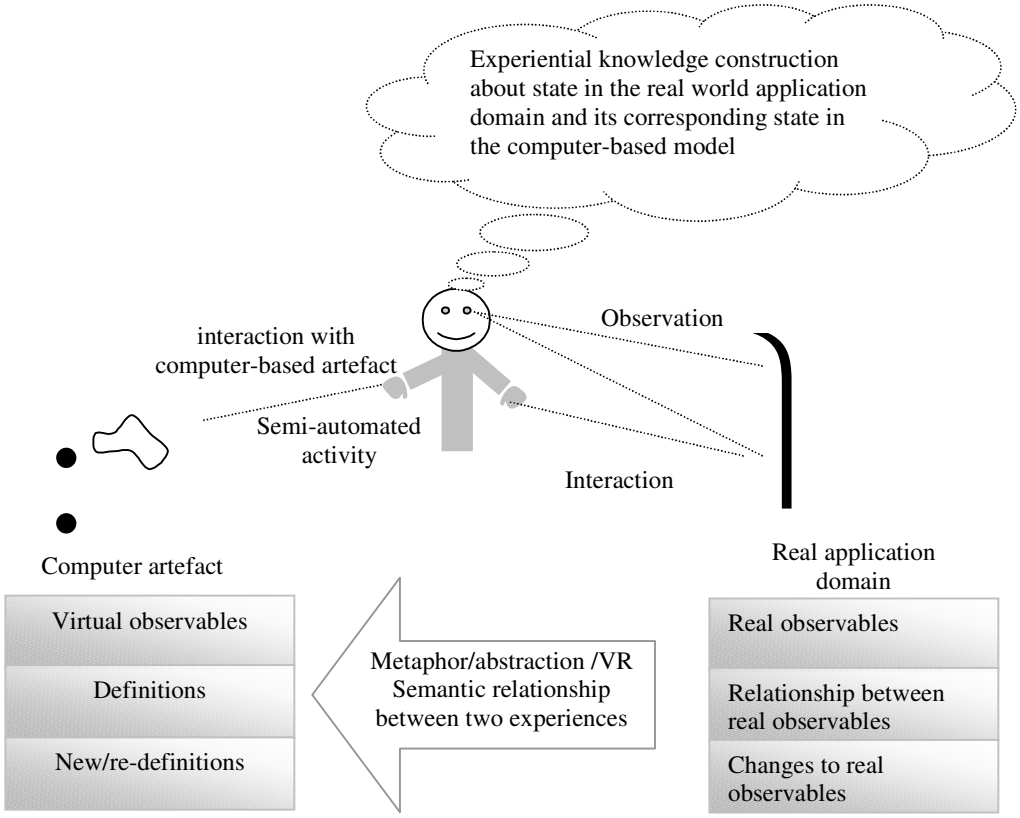

Figure 3.9 The semantic relationship between the computer-based artefact and its real world referent

[17] A feature provided by current EM tools.
[18] A research agenda for future generations of EM tools.

Figure 3.9 illustrates the concept of representing experiential knowledge about state in the application domain by experiential knowledge about state in a virtual environment (EM computer-based artefact). A semantic relationship is established between observables in the real world domain and the computer-based artefact. Observables in the computer-based model are not abstract representations of entities in the real world but reflect the modeller's understanding and interpretation of real world observables. Relationships between observables in the real world are associated with definitions relating observables in the computer-based artefact. Changes to observables in the real world application domain correspond to new definitions or re-definitions relating observables in the computer-based model.

In EM, the construction of the artefact is informed by how the modeller understands the real-world situation surrounding the software system under development. Specifically, the artefact embodies the modeller's expectations concerning the agency that affects observables and dependencies between them ('the modeller's *construal*').

Technology support

Empirical Modelling technology suggests a broad foundation for computing that harnesses features of Virtual Reality, AI, and database technology to establish a virtual environment that is semantically related to its real world referent.

c) the use of artefact for experiential knowledge construction

Experience

Experience, literally defined in [rhyme] as 'the accumulation of knowledge or skill that results from direct participation in events or activities", constitutes a ground base for truth and knowledge. Gooding (1990) asserts that experimentation is a hallmark of scientific activity, and attaches a great importance to the role of observation and human agency in practice.

Three issues are important in considering human experience in the real world:

- Acquiring experience
- Acknowledging an experience (being aware of the experience, and understanding it)
- Describing and sharing experience
- Learning from experience (personal subjective, public objective).

Knowledge

Knowledge is the psychological result of perception and learning and reasoning and can be implicit or explicit. While preceding knowledge, experience is broader, and ill structured. Knowledge is the outcome of experience that can be repeated with a certain degree of faithfulness. Reliable personal experience is subsequently translated into objective and

structured knowledge that helps in identifying a reliable behaviour for a system solving problems in the application domain. Dienes (et al 1998) definition of knowledge indicates that knowledge is limited and concise, whereas experience is not limited in space and time. Experience is subjective, and human imagination, observation, and agency are paramount. Personal experiential knowledge is subjective, ill structured, incomplete, and continuously evolving.

Common practice for knowledge construction

In common practice, constructing experiential knowledge of a particular domain can be initiated by adopting one of the two approaches: (i) empirical testing of theories pertaining to the domain; (ii) following personal insights in observing and experimenting within the domain

Both approaches are viable and support experiential knowledge construction to some extent. The problem with the first approach is the validity of the underlying assumptions of the theories in all possible situations. The second approach might be considered a primitive one, and is likely to be adopted by the non-expert in the domain. Reconciling the two approaches is a challenging task, and requires a framework that encompasses the experimentation, the formulation, the testing, and the amendment of theories. The first challenge is that the theory is represented abstractly and admits no easy amendment. The second challenge is devising a computational framework that supports a broad activity that embraces theory construction, validation, and testing. The third challenge is the continuous change that makes theory formulation hardly possible.

Experiential knowledge construction in EM

EM aims at supporting experiential knowledge construction that can potentially lead to knowledge formulation following continuous interaction with the computer-based model and the discovery of a reliable repetitive pattern of interaction.

Traditional use of the computer has focused on representing experience in the real world through formal approaches. However, EM technology relies on the key concepts of observation, agency, dependency, agent oriented analysis, and definitive representation of state and state transition. It adopts the techniques of construing a situation, constructing Interactive Situation Models (ISMs), metaphorically representing state through ISMs, and developing cognitive artefacts – typically computer-based. These key concepts and techniques, enacted using notations and computer tools, promise to deliver a framework (typically computer-based) that favours conducting experiments and gaining experiential knowledge of the real world. This framework aims at establishing a strong correspondence between real world activity and computer-based activity. Given the primary role of experience in knowledge construction and formulation, providing computer-based support to

this real world activity serves to enrich an initial, yet essential, phase in computer activity referred to as requirements engineering. Research work on EM technology to date has served as proof of concept for supporting an ongoing requirements engineering activity that gradually and continuously feeds all other derived computer-based activities.

d) Empirical Modelling technology enables the communication of definitive scripts to support collaborative distributed modelling

Motivation

Modelling the real world as seen in the eyes of an external all-powerful human modeller has many drawbacks. These stem from several factors: i. the individual bias in the modeller's understanding and interpretation of phenomena in the real world; ii. the load on the modeller who is supposed to play the role of all agencies affecting the state of the model; iii. the lack of realism in the modelling activity; and iv. the foregone benefit of group social activity in modelling.

Collaborative modelling in EM

The Distributed Empirical Modelling framework aims at overcoming personal modelling by supporting collaborative[19] distributed modelling. This can potentially serve the objectives of:

- redressing the individual bias in the modelling activity
- restoring the balance in the modelling activity by inviting every agent (human and / or automatic) to take his/her/its role in the modelling activity through appropriate views and privileges for actions
- bringing more realism to the modelling activity by involving every participant in the real world domain (user / developer / designer in the context of software system development or manager / personnel in the context of the business or financial enterprise)
- benefit from sharing insight and understanding in group social computer-based modelling

Empirical Modelling technology promises to support a situated group social activity by accommodating non-preconceived modes of interaction and taking account of the context and situation of the group social activity [Sun99].

The Distributed Empirical Modelling Framework

The distributed Empirical Modelling framework (DEM) as introduced in [Sun99] supports a modelling activity where 'the system modelled can be observed from the perspective of its component agents and an objective viewpoint or mode of observation to account for the

[19] Sun (et al, 1999) compares collaborative distributed activity to co-ordinative and subordinative. Collaborative distributed activity is situated and favour sharing insight in an open ended way.

corporate effect of agent interaction is identified". The DEM supports collaborative relationships between modellers concerned with understanding that is socially distributed [SB99]. Such relationships engage with issues of subjectivity and objectivity associated with distributed cognition [Hut95] and common knowledge [Cro94, EM87]. In a collaborative relationship, there is no possibility of relying entirely upon closed-world representation and preconceived patterns of interaction [SB98]. Such interaction is situated intelligently and can only be planned in advance to a limited degree.

The aims of DEM as introduced in [BS99] are to examine the relationship between communication media technology and human communication and to make more effective use of telecommunications technology in sharing and distributing cognitive models.

3.3 Conclusion

This chapter introduced key concepts, techniques, notations and tools in EM. Four illustrative case studies were considered in this introduction: the OLS regression to illustrate the use of definitive scripts in EM; the CAPM to illustrate the generalization of the spreadsheet concept in EM; the story of a retail trade in NYSE to illustrate the use of ISM and LSD notation; and the distributed stock market game to illustrate distributed EM. The distinctive qualities of model building in EM were highlighted. These include the focus on state as experienced; the maintenance of a semantic relationship between the EM model and its real world referent; experiential knowledge construction; and the use of definitive scripts in collaborative distributed modelling. These qualities give EM the potential to meet the technical and strategic demands for the wider agenda of computing that were introduced in chapter 2. This will be discussed in the following chapter.