Chapter 4

Distributed Empirical Modelling

A prevalent view in modern industrial societies is that group activities are an economically necessary and efficient means of production, and that such societies could not survive on the basis of individual effort alone [Smi97]. Group activities are particularly important in developing information systems, since this task is bound to involve the co-operative effort of users, developers and designers [AC98]. Any approach for supporting this social process should therefore take group activities into account. On the other hand, though EM has promising potential for software system development (SSD) as discussed in Chapter 3, the practical work of EM so far has focused on the role of the individual[1] modeller. In a typical application, the modeller, in the role of the only external observer, builds up a computer model for SSD by modelling. The model often converges to a state that represents the cognitive insights of the modeller. In order to apply EM to SSD, convenient support for group activities, in particular group interaction, should be provided. To achieve this, it is necessary to clarify the distributed perspective on EM and enhance the framework of EM to create a distributed modelling environment where interpersonal interaction between modellers (such as exploring shared knowledge and shaping agency of agents) can be effectively carried out.

[1] The emphasis in EM until now has been on groups of people creating a single model, or creating individual models that are not closely integrated.

4.0 Overview

This chapter aims to establish a framework for distributed Empirical Modelling[2] (DEM) by drawing on two important theories in social science: *distributed cognition* [Hut95] and *ethnomethodology* [Gar67]. Two topics associated with shaping agency and developmental strategy for SSD within DEM are also discussed.

The first section seeks to explain why - from both a practical and theoretical point of view – the distributed perspective on EM needs to be highlighted. In practical terms, these factors are considered: alleviating the overloaded responsibility of the modeller in EM, adjusting a critical bias with reference to the balance of subjectivity and objectivity in EM, catching up with the trend towards techniques for developing distributed software systems and providing an alternative to the traditional communication between users and developers. In the theoretical analysis which follows, the concept of distributed cognition is introduced to explain that meanings of group work are distributed across all participants rather than only inside any individual [Hut95]. Next, the concept of ethnomethodology [Gar67] is elaborated. It highlights the inadequacy of investigating the interaction between members of society simply from the viewpoint of an external observer. Both concepts indicate that the modelling activities for developing multi-agent systems should be invoked not only from the system level but also from the component level. This fact motivates the construction of a framework for DEM.

In Section 4.2, a framework for DEM is established. An approach, called E-modelling and based on a familiar approach to concurrent engineering, is first considered. Although modellers can perform E-modelling activities from both the system and the component perspectives in a distributed environment, they act as *external* observers so

[2] The term 'DEM' is used in this thesis for highlighting the distributed perspective on EM. It should not be misunderstood as if it is different from EM.

that the application of EM principles becomes more obscure. This is because the context in which each modeller interacts with the computer model is separated from the context in which he/she interacts with other modellers. Another approach called I-modelling is then proposed to highlight not only the distributed perspective on EM but also the principles and concepts of EM. New modellers as internal observers, called A-modellers, are introduced to shape the agency of agents in agents' customary context. This is reached by 'pretend play' whereby modellers act as agents to interact with each other. At the same time, a modeller called the S-modeller is involved. The S-modeller as the super agent shifts his/her super-agency from shaping the agency of agents to creating the context for agent interaction. With the involvement of the S-modeller and A-modellers, the framework that integrates the cognitive processes of modellers with their social process is constructed for DEM. This section ends with a discussion of the collaborative relationship between modellers. The relationship between modellers is analogous to that of participants in Gruber and Sedl's shadow-box experiment, where observers and experimenter collaboratively work together to construct a consensus between them [Goo90].

Section 4.3 focuses on distinguishing DEM from AI and previous variants of EM where shaping agency is concerned. First of all, the popular definitions of *agent* and *agency* in the AI field are briefly reviewed and their literal meanings (based on dictionary definitions) are explained. Next, the definitions and features of agent and agency in DEM are discussed. Within DEM, it is convenient to adopt the convention that an observable becomes an agent as soon as the modeller attributes a particular state change to it, and reverts to an observable when its agency disappears. With this interpretation of agent and agency, DEM highlights the fact that shaping agency is a cognitive process but also a social process where modellers interact with each other by acting as agents. Through the combination of both social and cognitive processes, modellers can adapt the agency of agents to change practice and at the same time make it accountable to each other (in the sense introduced in ethnomethodology).

Three strategies for SSD are discussed in Section 4.4. The *intentional* strategy, proposed by Dennett in [Den87] and underlying most conventional design models, is the strategy most commonly used by the developer of a software system. Through this strategy, the intentionality of the developer is embedded in the behaviour of the developed system. To complement the intentional strategy, the *experimental* strategy whereby the developer explores the system behaviour through exploratory experimentation is introduced. When both strategies are combined, the developed system reflects not only the intentionality but also the experimental experience of the developer.

Intentional and experimental strategies put the central focus on the developer, which gives rise to a gap between the developer's system and the user's system. *Evolutionary* strategies for SSD have been proposed as a way to narrow the gap, but these are traditionally developer-centred. DEM attempts to bridge this gap in a more significant way by adopting an evolutionary strategy that involves users. It treats a software system as an evolving system whose behaviour is adapted to its rapidly changing environment in an interactive and situated manner. In this way, the task of system development is no longer an exclusive prerogative of developers. Instead, users, based on their professional background and contextual needs, are also amongst the most important 'developers' of their system as it operates in the real world.

4.1 The Need for DEM

Research into supporting a group activity in EM stems from the need to develop multi-agent systems [ABCY94, BR94, BNOS90]. In [Bey97], two scenarios where EM are applied to develop a multi-agent system are described:

- **Scenario 1** (called *S1-modelling*): the modelling activity is centred around *an external observer who can examine the system behaviour*, but has to identify the component agents and infer or construct profiles for their interaction.
- **Scenario 2** (called *S2-modelling*): *the system can be observed from the perspectives of its component agents*, but an objective viewpoint or mode of observation to account for the corporate effect of their interaction is to be identified.

Due to the lack of a suitable tool that can support S2-modelling in EM, the main focus of EM in previous research work and most case studies has been only on S1-modelling[3]. Although S1-modelling can be used alone for the development of multi-agent systems, S2-modelling – from both practical and theoretical perspectives – is more useful for developing such systems. Four reasons for this are discussed here: the need to decentralise the modelling activity, to redress individual bias, to develop distributed systems, and to support interpersonal interaction.

Decentralisation of the modelling activity: In practical terms, S1-modelling empowers one modeller as the super agent to step into the computer model in order to explore unknown territory. It is fair to say that after accomplishing the modelling task the

[3] A variant of S2-modelling is found in some case studies (such as in [ABCY94, BR94]). With this variant, each modeller at the lower level can concurrently describe the system behaviour from the perspectives of component agents by using LSD, and a high-level modeller (that is, the super agent) performs the modelling activity. Since the lower-level modellers do not enact EM in a typical fashion, this variant is not regarded as S2-modelling in this thesis.

modeller should become an expert in the specific domain. However, this also means that the modeller must be capable of deriving sensible meanings from the model. For example, in modelling a vehicle cruise control system [BR94], there could be a number of roles for the modeller to play for the purpose of interacting with the computer model. In the role of a car engineer, the modeller should be aware that a transducer on a wheel emits one pulse per revolution; the speed is measured by a counter/timer that estimates pulse-rate. At the same time, being in the role of a driver, the modeller should be aware of the relationship between the operation of setting the cruise speed and the output power of a car engine.

In other words, the modeller in S1-modelling is responsible for playing various roles from different viewpoints in order to promote the enaction of EM. The magnitude of this responsibility leads to overloading, with implications for cost-benefit and productivity, when the modelled subject is very complex and involves a variety of knowledge. To overcome this problem of overloading, it is necessary to decentralise the modelling task from the sole modeller to a team.

Redressing individual bias: Gruber and Sehl's shadow-box experiment reveals the bias of an individual's perception and the advantage of teamwork [Goo90]. This experiment highlights how easily different observers can have different perceptions of a single situation, as illustrated in Figure 4-1. It also indicates that the individual's perception is often informed only by a part of the whole subject. In such a situation, according to Gruber and Sehl's explanation, observers have to communicate co-operatively with each other to identify a mutually agreed possible object hidden in the box. They exchange tentative constructs – or *construals* – of their personal experience in order to make more sense of the hidden object. A social process that moves private perception toward public consensus in the light of interaction with others is then commenced for construing or re-constructing each observer's own experience. In short,

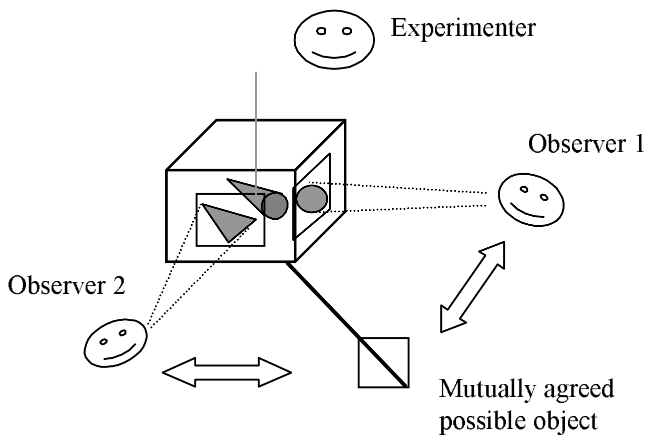


Figure 4-1. Gruber and Sehl's shadow-box experiment
(from [Goo90, p. 22])

this experiment highlights the fact that cognition based on the individual is very likely biased due to different situations and personal construals. To overcome this bias, it is necessary to integrate a social process with a cognitive process in order to construct a public, objective and universal construal between individuals. From this perspective, it is often not appropriate to model a multi-agent system by using S1-modelling alone.

The need to develop distributed systems: Distributed systems are increasingly popular in SSD. The splendour of centralised processing systems is fading away from the area of SSD because of up-to-date information technology and ubiquitous network communication. Instead, distributed systems that support a multi-user environment now appear to hold the spotlight. This new trend towards developing distributed software systems requires the support of a distributed modelling environment. In other words, if the intention is to apply EM to SSD, as explained in Section 3.3, then S2-modelling must be supported.

Support for interpersonal interaction: A very important motivation of highlighting S2-modelling comes from the research of D. Sonnenwald on 'communication in design' [Son93]. Sonnenwald pointed out that many design situations include a number of participants who need to interact with each other for building up an evolving artefact to support patterns of work activities, social groups and personal beliefs. Indeed, the importance of interpersonal interaction and of mutual understanding between all participants in SSD has been widely recognised [Bos89, Sal87, VF87, VPC98]. The development of a software system is not regarded as an act of individual creation but instead as the product of social interaction in a community [AC98, Flo87, XIA98]. Information arising from the interpersonal interaction between the user and the developer forms the foundation of SSD, and is therefore the key factor in determining the success or failure of a SSD project. For such interpersonal interaction for knowledge exploration and creation in SSD, it is obvious that S2-modelling can provide more support than S1-modelling.

At the same time, Sonnenwald's research also concludes that a collaborative tool is needed to support the mutual exploration of participants in order to facilitate interpersonal interaction and the creation of knowledge amongst participants. However, support for interpersonal interaction in SSD is limited and restricted to the use of traditional methods, for example, paper documents and conversational dialogue. Paper is passive and can only serve as a repository for information [Fis91, DS97]. Dialogue is liable to encounter those so called 'within', 'among' and 'between' communication obstacles[4] [VF87]. Although Sonnenwald gives few details about the nature of the collaborative tools that she has in mind, a computer-based system should be very promising in comparison with traditional methods[5]. This is because a computer-based system can be interactive and can facilitate human agents in searching, understanding and creating knowledge in a significant way [Cla97, Fis91, HM96, Rei97, FP88]. In this respect, with the features explained in Chapter 3, EM has proved its significant merits in improving an individual's

[4] The 'within' obstacles are those cognitive and behavioural limitations within the individual. The 'among' obstacles arise when participants have inconsistent or conflict viewpoints on the same thing. The 'between' obstacles are then mainly due to the fact that the involved parties speak two different languages.

[5] The computer-based interpersonal interaction should not be regarded as a replacement for conventional means of communication but as an important enhancement of existing means for communication.

understanding, creativity and learning [Bey97, Bey98, Nes97, Rus97, SB98]. In order to support knowledge exploration and creation between participants, the distributed perspective on EM needs to be clarified and highlighted in order to enable component agents to collaboratively interact with each other. In other words, it is necessary to provide support for S2-modelling.

Apart from these practical needs, an ontological debate that is associated with the use of S1-modelling and S2-modelling also indicates that S2-modelling is needed for developing multi-agent systems. With S1-modelling, the sole modeller as the super agent is empowered to interact with his/her computer-based model and the referent in the real world in order to explore, expand and experience his/her cognitive states. This modelling process is concerned with the construal of phenomena, empirical evidence and immediate experience, and is characterised by commitments based on the modeller's experience and knowledge [Bey98, Bey94, Rus97]. Whatever the system through which EM is enacted, the interaction of agents is viewed as a phenomenon experienced (or known) by the modeller and is construed in his/her particular context. Hence, the agency of agents (that is, the interaction between agents) is conceived in the privacy of the modeller's mind (cf. [BeyMSc, Bey98]). This inevitably amounts to a private, subjective and provisional interpretation of the modeller, that is, a first-person perspective.

In [Bey98], Beynon explains that this first-person perspective can be projected to the second- and third-person perspectives in order to build up common experience and public knowledge. The essential principle behind the projection of agency from first to second person is that "through experiment and perception of dependency I can identify families of observables (constituting a 'you') who can be construed as acting in ways congruent to my own" [Bey98]. And, from a third person perspective, an observable must be an element of "our experience that empirically appear to be common to all other human agents subject to what is deemed to be the norm" [Bey98]. Hence, Beynon argues that “objectivity is empirically shaped concurrently by our private experience, and our experience of other people’s responses [in a common environment]” [Bey98]. S2-modelling is crucial if this process of projection of agency is to be empirically investigated.

Without S2-modelling, ontological problems obstruct any insight into the projection of agency from first to second and third person. The first-person perspective, that focuses only on private experience of the modeller as an external observer, cannot account for the interpersonal interaction between agents that involves aspects independent of private experience. The second-person perspective raises the issue of how an identified agent (constituting a ‘you’) – in particular if this ‘you’ is another person – can be deemed to gain experience and knowledge in the common environment that is congruent to the modeller’s own. Similarly, the third-person perspective raises the issue of how interpersonal interaction that is inherently unpredictable can be construed as ‘persons following standards or rules without referring to their own circumstances and contexts’.

Two important theories in social science are very helpful in clarifying the above issues: distributed cognition [Hut95] and ethnomethodology [Gar67]. The concept of *distributed cognition*, proposed by E. Hutchins in [Hut95], represents a synthesis of cognitive, anthropological and social scientific approaches to the study of collaborative work. Its central theme involves “locating cognitive activity in context, where context is not a fixed set of surrounding conditions but a wider dynamical process of which the cognition of an individual is only a part” [Hut95, p. xiii]. Hutchins argues that interpersonal interaction should be seen as an activity that is undertaken in social settings by using various kinds of tools rather than as a solitary mental activity. His concern is that overemphasis on representing or describing a ‘knowledge structure’ that is somewhere ‘inside’ the individual overlooks the fact that interpersonal interaction is always located in an intricate social world from which it cannot possibly escape. Hence, Hutchins emphasises that cognition for a group activity is a process socially distributed between individuals and artefacts, that is, "the group performing the cognitive task may have cognitive properties that differ from the cognitive properties of any individual" [Hut95, p.176]. Any analysis of such a group activity thus has to account for the interaction between participants. This account softens the traditional boundary between individuals and their environment, in which individuals are separated from their environment and at the same time each individual is isolated from others.

Ethnomethodology[6] is "the empirical investigation ('-ology') of the methods ('method-') people ('ethno-') use to make sense of and at the same time accomplish communication, decision making, reasonableness, and action in everyday life" [Rog83, p. 84]. In his major work, H. Garfinkel, the originator and guiding figure of this approach, indicated that ethnomethodology concerns "practical activities, practical circumstance, and practical sociological reasoning as topics of empirical study" [Gar67, p1]. He argued that social actors are, through their own actions, unavoidably engaged in producing and reproducing the intelligible characteristics of their own circumstances [Gar67, p23]. It is therefore not satisfactory to describe or interpret their circumstances by reference to rules or algorithms which are external to, or independent of, the ways in which its characteristics are recognised, used and produced through practical actions in a contingent manner. In this sense, Garfinkel affirmed that "social facts are the accomplishment of the members" [GS70, p.353, quoted in [Cou95]]. Social actors embody those rules of social reality in the process of their everyday social practices. In the process, their concrete activities give sense to their surrounding world and naturally exhibit the social competence that affiliates themselves with this society, allowing them to be recognised and accepted [Cou95]. In other words, ethnomethodologists do not want

[6] A very wide range of issues has been discussed in ethnomethodology, such as how the feeling and emotion of a human agent in everyday life affect his/her behaviour [AA87]. Only those aspects that are described in this paragraph are taken into account in this thesis.

to import any of their own assumptions of social facts as objects into their descriptions. By contrast, they make social facts observable and capable of being described and constituted through naturally organised ordinary activities. Through this process of reflexive constitution, social facts are accomplished in the ongoing process of the interaction between members. In addition to studies of social problems, such as education, medical practices and organisational processes, the concept of ethnomethodology has been applied to scientific research, for example, to the work of mathematicians on mathematical proofs [Liv87].

According to the concepts of distributed cognition and ethnomethodology, the projection of agency from first to second and third person hence has to take sufficient account of the social dimension of understanding and the commitment it entails. T. Winograd and F. Flores argues that it is very important to shift knowledge and understanding from an individual-centred conception to one that is socially based:

> Knowledge and understanding (in both the cognitive and linguistic senses) do not result from formal operations on mental representations of an objectively existing world. Rather, they arise from the individual's committed participation in mutually oriented patterns of behaviour that are embedded in a socially shared background of concerns, actions, and beliefs. This shift from an individual to a social perspective – from mental representation to patterned interaction – permits language and cognition to merge [WF86, p.78].

In other words, cognitive properties that are distributed across members should refer not only to each individual cognition (a cognitive process) but also, more importantly, to the interaction between all individuals and their environments (a social process) [Hut95, WFH96].

The ontological issues raised by the projection of agency, and the relationship between the S1-modelling and the S2-modelling perspectives, are familiar in sociology.

For instance, Trigg (cf. [Tri85]) refers to ‘a vexed issue in the social sciences’: whether there is something to be discovered at the social level or whether everything of significance can be dealt with at the level of the individual. The discussion of this sociological problem is beyond the scope of this thesis. The emphasis given here is on reconciling both global (that is, social) and local (that is, individual) perspectives on social interaction in a society [Tri85]. After all, most human-centred activities are so complex that both perspectives are intertwined and interdependent. In the same manner, it becomes apparent that S2-modelling and S1-modelling should be supported concurrently in order to reconcile the global and local perspectives on the development of a multi-agent system [Bey97].

The need to support both S1- and S2-modelling concurrently highlights distributed perspectives on EM that operate in two different dimensions, and are embedded in both kinds of modelling activity. In the ‘vertical’ dimension, individual modellers that perform modelling activities from the system level and the component agent level should be involved. In the ‘horizontal’ dimension, modellers that observe the system from the perspectives of different component agents are also needed.

4.2 A Framework for DEM

This section aims to construct a framework in which not only S1-modelling but also S2-modelling can be supported concurrently. A framework for distributed Empirical Modelling (DEM) that highlights the distributed perspective on EM (described in the last section) is established in the first subsection. Two methods for constructing this framework are considered and examined. In the second subsection, the collaborative interaction between participants is discussed. This is very important for SSD (relevant discussions are given in Section 4.3 and 4.4) and requirements development (discussed in Chapter 7).

4.2.1 Constructing the Framework of DEM

From the distributed perspective on EM, modellers for both S1- and S2-modelling activities at the system level and at the component agent level should be involved. At the system level, the modeller, called the *s-modeller* (the small s is used to denote its role as the super agent), is involved for enacting S1-modelling in order to examine the whole system behaviour (as shown in Figure 4-2). As explained in Section 2.2, the s-modeller

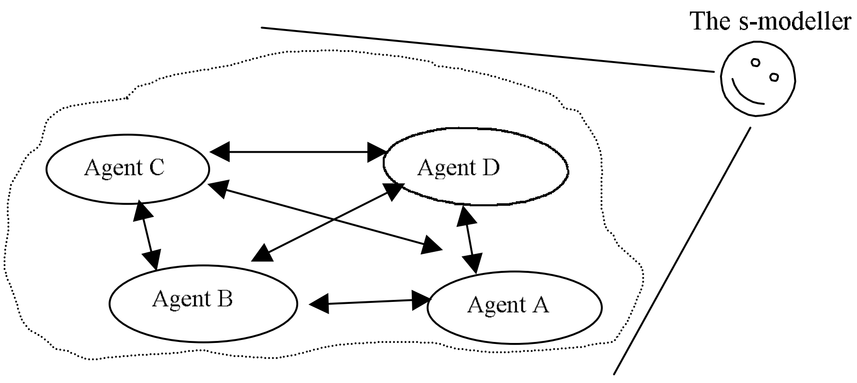


Figure 4-2 The relationship between the s-modeller and agents (the sole modeller in the system level)

has super agency to attribute the privilege of state change to agents and intervene in the interaction between agents.

In addition to the s-modeller as the super agent, modellers (called *a-modellers*) that can perform S2-modelling activity from the perspectives of component agents are also involved. Unlike S1-modelling, that is to a considerable extent understood, S2-modelling has thus not been explicitly discussed in previous work in EM[7], due to the lack of supporting tools. For instance, insufficient attention has been given to how a-modellers enact S2-modelling activity in a distributed environment in terms of observables, dependency, agents and agency.

To motivate DEM as introduced in this thesis, previous work on applying EM to S1- and S2-modelling is first discussed. This is based on an approach that is often used in concurrent engineering for decomposing a complex task to several small parts to be shared by a group people working together [DS94, Yeh92]. Within this approach, called *E-modelling*, each modeller is an *external* observer who observes the system as if from

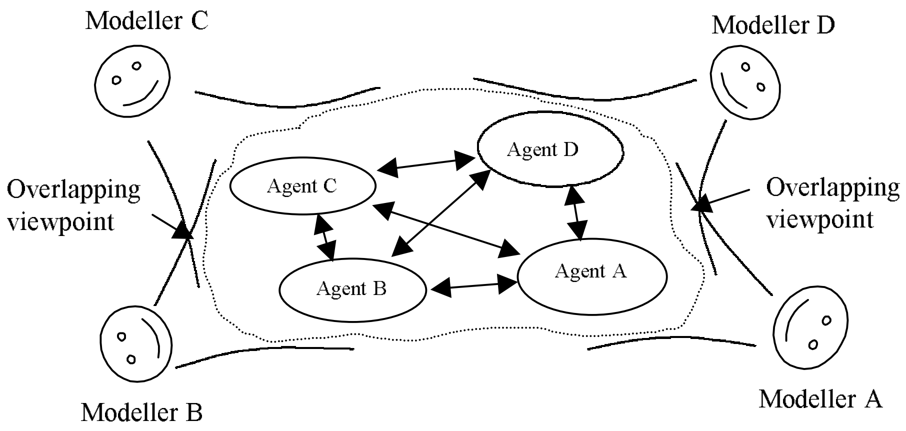
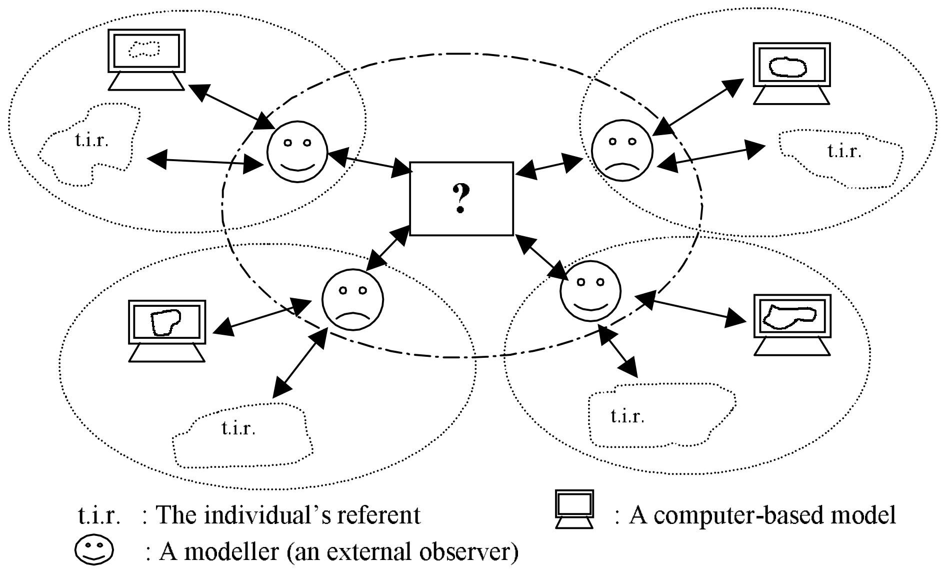


Figure 4-3 The relationship between a-modellers and agents (modellers in the component agent level)

[7] For example, it is not clear if all modellers use a shared computer model or each of them has his/her own computer model. For convenience, it is assumed in this thesis that each modeller has his/her own computer model and can enact EM in ways that are described in Section 2.2.

the perspective of a particular agent (as shown in Figure 4-3) and can enact EM individually (cf. [ABCY94, BR94, BNOS90]). For example, for developing a vehicle cruise control system (VCCS), a-modellers might be a dashboard designer and a car



t.i.r.
t.i.r.
?
t.i.r.
t.i.r.
t.i.r. : The individual's referent
: A computer-based model
: A modeller (an external observer)

Figure 4-4. A framework for DEM based on E-modelling

engine designer who observe the system to take account of the perspectives of the car driver and the car engine respectively (cf. [BBY92]). This leads to a framework that supports – to some degree – both S1- and S2-modelling activities in a distributed environment (as illustrated in Figure 4-4).

Within the E-modelling framework, each modeller (including the s-modeller and a-modellers) can enact EM as an external observer by observing the system that is being developed from a particular context (that is, the s-modeller observes the whole system, but an a-modeller observes a part of the system). By enacting EM, each modeller interacts with his/her own computer model and his/her own referent in order to explore, expand and experience his/her own mental model. To enable these modellers to enact their modelling activities together, an agreement about naming and sharing common data, such as observables and definitions in EM, is often established. With this agreement, all modellers can interact with each other to shape their overlapping – but potentially conflicting or inconsistent – viewpoints, for example, through managerial negotiation or mediation. As a result, once the conflict between these computer models is eliminated through such social interaction, the intended multi-agent system could be obtained by integrating together all definitive scripts, for example, into the s-modeller's computer model[8].

Although this framework enables modellers to develop a multi-agent system by enacting both S1- and S2-modelling activities in a distributed environment, it does not provide modellers with sufficient support for their interaction. For example, inconsistencies or conflicts in shaping the agency of agents with the application are often eliminated through social activities between modellers, such as managerial negotiation and mediation. The involvement of many social interactions between modellers implies that the advantages of enacting EM will be largely reduced, since the context for modelling their computer models must be separated from the context in which they interact with each other. For example, modelling a situation through 'what if' experiments for the modeller from the component agent level will become very difficult and make little sense, since only a part of the system is taken into account. In other words, E-modelling activities do not accord well with the principles and the concepts of EM.

In addition, according to the concept of ethnomethodology, external observers find it difficult to understand the real context of an agent on the basis of their individual viewpoints. For example, the modeller for the driver in VCCS expects to have a digital

[8] In this framework, a communication network that connected the computer models of all modellers could be used to help information processing, for example, the collection and integration of definitive scripts from different computer models. From the perspective of modelling, the modelling activities between modellers are still conceptually independent. A broader use of a communication network will be given later.

speedometer, but the modeller for the speedometer designer in VCCS takes the normal type of speedometer into consideration. After mutual interaction, an agreement (for example, the need to build a digitalised speedometer) could be gained. However, when the driver actually interacts with this kind of speedometer, as when glancing at it while driving, he/she discovers that it is hard to read the displayed figure because of sunshine. Modellers who view the system as external observers can only make sense of the interaction between agents from their own context rather than from the contexts of the component agents themselves. After all, performing as an agent and being able to describe and analyse the methods used by an agent are not the same thing. Although external observers can change their roles in order to capture more information, each changed role as an external observer can only understand the interaction between agents by imposing his/her own context. Hence, from the perspective of ethnomethodology, it is not appropriate to account for the interaction between agents through E-modelling.

Accordingly, a framework based on E-modelling is not well suited for DEM, even though it can integrate S1- and S2-modelling together in a distributed environment for the development of multi-agent systems. A new approach (called *I-modelling*) that highlights not only the distributed perspective on EM but also the principles and concepts of EM is proposed by the author in this thesis (see Table 4-1 for a summary of the descriptions of S1-, S2-, I- and E- modelling). This approach introduces a distributed environment in which the roles of modellers in relation to component agents are both externalised and internalised. Through externalisation, the super-agency of an external observer in DEM is shifted from shaping the agency of agents to creating the context for agent interactions. Internalisation then enables modellers to improve their understanding by enacting the interaction between component agents in its natural context as internal observers.

In I-modelling, *internalisation* means shifting the role of a modeller from an external observer of the system inwards to that of an agent inside the system. By drawing on the concept of ethnomethodology [Gar67], new modellers at the agent level are introduced. Like modellers in E-modelling, these modellers are responsible for establishing the correspondences between their own computer models and referents to explore, expand and experience their mental models. But unlike modellers in E-modelling who are in the role of external observers, these modellers are placed in the role of agents within the application. In other words, each of these modellers, called an *A-modeller*[9], can act as an agent from the perspective of an *actor*, not only from the perspective of an observer.

S1-modelling	Modelling activity that is centred around the system behaviour	
S2-modelling	Modelling activity from the perspectives of component agents	
E-modelling	An approach to modelling a software system whereby modellers invoke S1 and S2 modelling activities by acting as external observers	
	s-modeller	The modeller who enacts S1-modelling
	a-modeller	A modeller who enacts S2-modelling
I-modelling	An approach proposed in this thesis that requires most modellers to model a software system by acting as internal observers	
	S-modeller	The modeller who creates diverse situations for A-modellers to reflect the different possible contexts for their interaction (this can be viewed as S1-modelling)
	A-modeller	A modeller who can act as an agent to shape the agency of that particular agent (in a sense, this activity can be regarded as S2-modelling)

Table 4-1 A summary of the descriptions of S1-, S2-, E- and I-modelling

As an actor, an A-modeller can pretend to be an agent within the application by enacting the ordinary interaction with other agents. This *pretend play* relies on an important belief in the ethnomethodology that each agent is capable of managing the world and of 'being-in-the-world' [Gar67]. Hence, A-modellers can either wait for

[9] The capital A is used to denote the role of the modeller in acting for an agent.

stimuli before they react or can autonomously trigger action to interact with others in their customary context, that is, from the viewpoint of agents themselves rather than from the viewpoint of the role of a particular external observer. In this way, A-modellers can get insight into the interaction between agents in the agents' context.

Through pretend play, A-modellers are regarded as *internal observers* rather than external observers, as shown in Figure 4-5. An external observer focuses on attributing state changes to a particular agent, but an internal observer pays more attention to appreciating the situations in which interactions between an agent and other agents occur. For an external observer, the interaction between agents is regarded as a form of state change of agents in the application system. The mechanism of stage change is often vindicated through observation and experiments from a specific viewpoint, e.g. an engine designer's viewpoint. Even though an external observer can 'experience' such an interaction by interacting with the computer model and/or the referent, this interaction is still based on a private and subjective perspective outside these agents, such as that of a designer or a user.

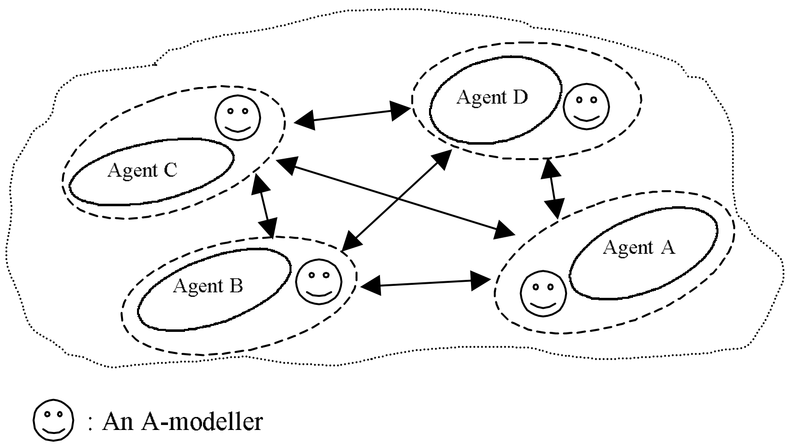


Figure 4-5 A-modellers acting as internal observers in a being-participant-observer way

In contrast, being an internal observer means interacting with other agents by acting as an agent. This kind of interaction that is exercised by internal observers has a strong contextual dependency, and can only be understood in a situation where agents are engaged in activities that they regularly and ordinarily perform. For example, in I-modelling for the VCCS, an A-modeller, who acts as the *car engine* agent rather than observing the system from the viewpoint of a car engine designer, can be introduced (cf. [BR94]). In comparison with a car engine designer acting as an A-modeller, such an A-modeller can gain much more insight into how the car engine interacts with other agents, such as the driver and the environment. By acting as a car engine, this A-modeller can wait for stimuli, such as those from the driver who starts the car engine or from its surrounding temperature, before responding. The A-modeller can also trigger action to interact with other agents, for example, by making itself break down due to unknown reasons. From the viewpoint of a car engine designer, these situations may not be important enough to be taken into account, but they are ordinary activities for a car engine. In particular, due to the uncertainty of incoming stimuli triggered by other agents, the context is usually situated beyond the imagination of a car engine designer.

This *being-participant-observer* way of acting as agents provides modellers themselves with important resources to account for their (agents') activities[10]. As the ethnomethodologist A. Coulon explained in [Cou95, p.23]: to describe a situation is to

[10] According to the way in which an observer participates in the activities of members, three membership roles have been proposed in [AA87]: peripheral, active and complete membership. The *peripheral* membership role allows the observer to participate in the activities of members without engaging in the most central activities. As an *active* member, the observer can participate in the core activities in much the same way as members, but he/she must hold back from committing themselves to the goals and values of members. With *complete* membership, the observer is expected to participate in the activities of members from the perspective of a full member. In I-modelling, A-modellers might adopt different membership roles that change over time for gaining different resources, but they become involved in complete membership roles for performing pretend play.

constitute it. Through this kind of reflexivity[11], the interaction between A-modellers can be conflated with the procedures that agents carry out in order to make their interaction accountable. In other words, by acting as agents to modelling situations, internal observers are able to describe the ways in which agents themselves make sense of their interaction and consequently constitute the reality of the interaction between agents.

From this perspective, the proposed interaction of agents through the enaction of EM in pretend play can enrich both the context of the agents' interaction and the mental model of the A-modeller. More significantly, when A-modellers interact with each other in the customary context of agents, they can observe, experience and account for the interaction between agents. This enables them to shape the agency of agents with reference to the interaction between A-modellers. This echoes W. Sharrock and G. Button's concept of 'acting as a member of a group to contribute positively to the creation of social interaction' [SB91].

The proposed pretend play raises a fundamental question: can an A-modeller, being a human agent, pretend to act as a non-human agent, such as a machine? An affirmative answer to this question is justified because the mechanism of a non-human agent is conceived in terms of human intentionality [Den87]. It is on this basis that the activities constituting the mechanism are prescribed, are well-defined and – more significantly – are made accountable to other agents. By appealing to such an account, it is natural for an A-modeller to engage in these activities in a mechanical manner. Even in the case of unpredictable activities, this accountability is still relevant. This is because it is plausible to attribute human uncertainty and unpredictability to a non-human agent. The attribution

[11] This is the feature of social order that "presupposes the conditions of its production and at the same time makes the act observable as an action of a recognisable sort" [Cou95, p.23]. It refers to "the equivalence between describing and producing interaction, between understanding and articulating the understanding" [Rog83, p.94]. For example, while people are talking, they are building up – at the same time that their words are uttered – the meaning of what they are saying.

of such failings to a non-human agent exactly highlights the possibility of its failure to confirm to the designated mechanism.

This way of attributing human-like mental features to machines is not a new concept. It has been broadly adopted in the development of AI systems [McC78, Sho93, WJ95, Rao94]. Most of them believe that to ascribe mental notions, such as beliefs, free will, intentions, abilities, autonomy, and so on, to a machine is legitimate and useful. More details will be given in the next section.

In practice, it is not necessary to involve as many A-modellers as there are agents when enacting their interaction. For instance, there may be no need to introduce an A-modeller for an agent whose interaction with others is entirely predictable. Such an agent can be temporarily regarded as a reliable machine that is responsible for specific state changes. In other words, the extent to which enaction by an A-modeller is appropriate depends largely on the character of the interaction between the agent to be enacted and other agents. An A-modeller can be introduced into or withdrawn from I-modelling at any moment in order to reflect the predictability of this agent's interaction with others.

In addition to the agents mentioned above, another very important 'agent' is taken into account in I-modelling. This agent is the rapidly changing environment of agents that is beyond the control of any agent. This omnipresent agent, that might for some reasons be regarded as 'God' [Fey75], can create contexts for agents in an open-ended manner in order to intervene in their interaction. This super intervention provides an effective way to express the uncertainty of agents themselves and their surroundings. By failing to take this into account, traditional rationalism restricted to rigid algorithms is hard to adapt itself to the emerging context in the *real* world [WF86, Suc87, Fit96].

For EM, super intervention is embedded in the modelling activities of the super agent, that is, the modeller. With this super-agency, the modeller can shape the agency of agents by directly attributing the privilege of state change to agents and performing 'what if' experiments. To enable A-modellers to shape the agency of agents in the agents' context, I-modelling *externalises* the relationship between the modeller (as the super agent) and agents by shifting the focus of the super-agency from shaping the agency to creating contexts for agent interaction. A modeller called the *S-modeller* is introduced in I-modelling. Even though the S-modeller still has the authority to perform the individual modelling task, more emphasis is placed on modelling the context of agents by creating diverse situations that can affect the interaction between A-modellers in the role of agents. In this way, the responsibility for modelling the agency of agents is largely shifted to the A-modellers, and this agency emerges from their interaction with each other.

The S-modeller, acting as the rapidly changing environment, can then create more various situations for A-modellers than A-modellers themselves can do. For example, a car engine may suddenly break down, or the driver may fall half-asleep. These situations, which are usually unexpected, enrich the context of the agents' interaction, and as a result improve the understanding of the S-modeller and A-modellers of the agency of agents. It should be noted that this improvement in I-modelling is reached by means of collaborative interaction between modellers (including A-modellers and the S-modeller) rather than the interaction between the S-modeller as the super agent and his/her individual computer-based model. More details of the collaborative interaction are given in the next subsection.

Network communication has an enabling role to play for I-modelling. The computer models of the S-modeller and A-modellers must be connected together in order to support the interaction between them in the form of pretend play. The society of agents is indivisible, though each agent may just be a part of this society. Each change triggered by an agent may cause effects on other agents so that the propagation of the change to others becomes very important. By using a communication network that connects all computer models together, this propagation can be achieved even though agents are separated in a distributed environment. With the aid of a communication network, a framework that integrates the S-modeller and A-modellers and supports S1- and S2-modelling concurrently in a distributed environment can hence be constructed. This framework (as illustrated in Figure 4-6) highlights not only the distributed perspective on EM (as described in the last section) but also the principles and the concepts of EM (as described in the Chapter 2). More technical details are discussed in the next chapter.

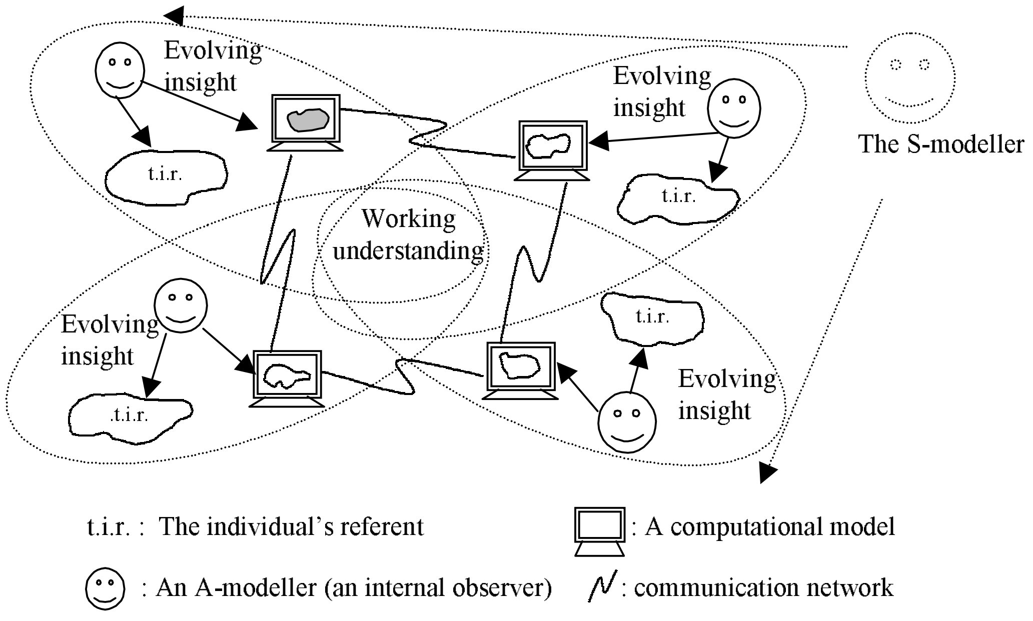


Figure 4-6 A framework for DEM based on I-modelling

The connection of modellers' computer models offers the additional advantage of improving the interpersonal interaction between modellers. The complexity and difficulty of interpersonal interaction in design involving participants from different groups have been revealed in [Son93]. Conversational dialogue on the basis of mental models is one of the easiest and most popular approaches to the interaction between human agents, but it does result in a number of interpersonal interaction problems, such as misunderstanding, confusion, and excessive time-consumption [Bos89, VPC98].

To overcome these problems, a lot of communication media, for example, formal notations, diagrams, and charts, have been devised to complement this social interaction in practice. However, most of these media, used as means of knowledge representation, are inactive and cannot support the social interaction efficiently enough to cope with dynamic change [DS97, Fis91]. Some computer models with static visualisation have been suggested for improving the method of human communication [LL94, LR91], but they can only provide limited help because they focus on knowledge representation.

In contrast to these models, the computer model with interactive visualisation has the potential to facilitate interpersonal interaction in an open-ended, interactive fashion. Models of this kind embodying individual construals are used in EM as a modelling medium [Rus97] and a knowledge construction medium (as described in Chapter 2). Their visualisation is obtained by representing the real-world referent using graphical metaphors. Since the internal representation of these graphics is expressed in the form of definitive scripts, any change in these scripts can be immediately propagated to their higher-level representation, viz. the graphics themselves. That is to say, via the communication network, the visualisation can interactively reflect the interaction between agents. Once a computer model is changed, this change can be propagated to other computer models, and this in turn will affect their modellers. By such interaction mediated by visualisation, the interpersonal interaction between modellers can potentially be improved[12]. In this respect, these computer models promote interpersonal interaction amongst modellers and serve as 'equipment for language' in the sense highlighted by T. Winograd and F. Flores [WF86, p.79].

[12] Due to the limitation of the author's research time and the lack of suitable projects, this improvement has not yet been evaluated in practice. An evaluation of the impact of computer-mediated interpersonal interaction would be helpful and is proposed as future work (see Section 8.3).

4.2.2 The Collaborative Relationship between Modellers in DEM

The proposed way of applying the ethnomethodological method to construct interaction between agents from an empirical environment is subtle. Clearly, the progress of this approach relies very much on the modellers involved. To illuminate the collaborative relationship between modellers, it is helpful to review the Gruber and Sehl shadow-box experiment described in the previous section.

Within this experiment, observers construe personal experience by simultaneously interacting with their own internal mental model and external environments in a way discussed earlier (Section 3.2). Both kinds of interactions are important resources for observers in making sense of the projection seen and the object hidden. Through both, observers can not only construct their own individual local knowledge but also contribute to the emergence of the distributed global knowledge amongst them. The former is associated with the projection seen and reports of others' construals. The latter, underlying the shared understanding of observers, is concerned with the agreed object that is compatible with the imaged objects of observers. It is not inside any observer's individual minds, but rather it emerges from distributed cognition across observers. That is to say, given the ability to interact with each other, observers collaboratively work together in order to make progress in constructing both the individual local knowledge and the distributed global knowledge.

In fact, another particularly vital participant in the shadow-box experiment is the experimenter, although he/she receives very little attention. It is the experimenter who sets up the context of experimentation for observers in order to examine the construals of observers. The experimenter is empowered to intervene with observers' contexts by altering the seen projection, for example, by changing the orientation of the hidden object. This intervention serves to provide observers with diverse resources to refine their construals, and this in turn leads to the reconstruction of both local and global knowledge mentioned above. In other words, the experimenter performs the role of a resources-for-knowledge-construction supplier and thus enables the observers to construct both kinds of knowledge. With the participation of the experimenter, the construction of both kinds of knowledge can be more effective and efficient, though it also depends on the experimenter's ability to create the context for observers. Therefore, the importance of the experimenter in this process of knowledge construction should not be underestimated.

On the other hand, the experimenter is also a constructor of knowledge, just like any observer. As described above, through the context created by the experimenter, observers are able to construct or reconstruct their knowledge and also to enrich their own contexts with surprising discoveries. This enriched context in turn provides the experimenter with useful resources with which to construct the individual global knowledge about this experiment. Theoretically, this global knowledge can include anything relating to the experiment, but usually its focus is on the experimenter's insight into the whole experiment. Its content could be very high-level and multifaceted. For example, the knowledge can concern how the new context affects observers' construals and what object, compatible with individual perceptions, is agreed by observers. These concerns are relevant to the experiment itself, but are also subject to the egocentric perspective of the experimenter.

Clearly, even though this global knowledge is always private and subject to the experiment, it indeed provides observers with situational contexts and as a result hopefully enriches both kinds of knowledge at the observer level, that is, the individual local knowledge and the distributed global knowledge. In most cases, the more situations that the experimenter creates, the more resources are provided to observers and the more feedback the experimenter obtains. This fact indicates the symbiotic relationship between the experimenter and the observers in constructing knowledge.

The way in which the A-modellers and the S-modeller work together collaboratively within DEM is much the same as the way in which the experimenter and the observers work together in the shadow-box experiment. Within DEM, A-modellers acting as agents can interact with each other to construct distributed cognition amongst them as described above. For example, an A-modeller can be introduced to act as a driver who interacts with both a brake and an accelerator to control the required power supply of a car engine. Given the visualisation of his/her computer model as shown in Figure 4-7,

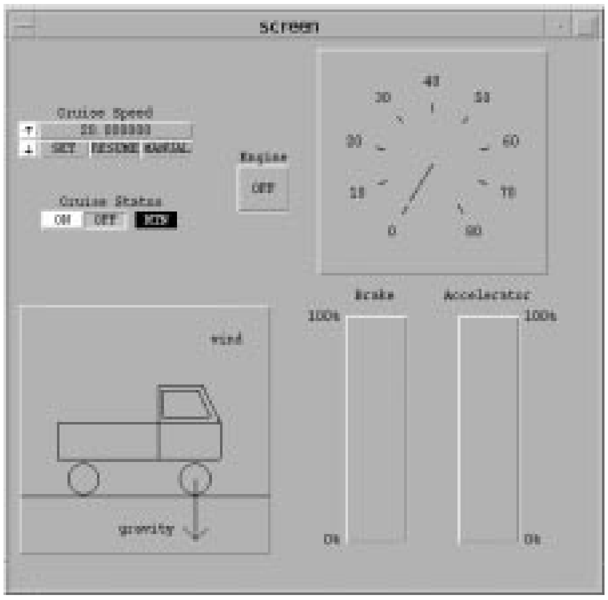


Figure 4-7. A visualisation of a vehicle cruise control system

the A-modeller can click on the box representing the accelerator to request the car engine to increase its power output. In this case, another A-modeller acting as a car engine could respond to this request in any chosen fashion, for example, by complying with the request, or making no response at all. In this way, each interaction between A-modellers becomes a constitutive part of the interaction between agents.

In other words, through interaction between modellers that takes the same form as that occurring in the ordinary life of agents, distributed cognition among A-modellers –

and also in the similar manner among agents – can be established reflexively. The difference between the observers in the shadow-box experiment and A-modellers is that the latter interact with each other through computer models connected together via a network rather than through linguistic dialogue[13] (the resulting implications for human communication have been explained in the previous section). Connecting computer models in this fashion makes individual contexts combine with each other to create a social context corresponding to Hutchins's model of what actually happens in a human society [Hut95].

At the same time, the S-modeller, like the experimenter in the shadow-box experiment, can intervene in the A-modellers' context to enrich not only his/her own global knowledge but also the local knowledge of each A-modeller and the distributed knowledge of A-modellers. The computer model of the S-modeller is established by integrating the computer models of all A-modellers. This integration is in accordance with the idea that the experimenter has a global view that embraces all components in the experiment. This view reflects the distributed cognition amongst agents, and describes how this is constituted through the interaction of A-modellers.

When the S-modeller intervenes in the computer model of any A-modeller, the new context can be propagated to other computer models through dependency maintenance and the network connection. This propagation can thus effect a change to these models and give rise to further interaction between A-modellers. As a result, the situational context of agents can be described and constituted through these interactions. The influences of the S-modeller in the situational context of agents should not be overlooked. It should be noted that the individual global knowledge of the S-modeller is not

[13] This should not be misunderstood as proposing that conventional communication between human agents can be completely replaced by the interaction between these connected computer models.

necessarily identical with the distributed knowledge of A-modellers, though both are closely interrelated.

In summary, the relationship between the S-modeller and the A-modellers in DEM is on par with that between the experimenter and the observers in the shadow-box experiment. On the one hand, all modellers construct their own individual knowledge by interacting with their own computer models. On the other hand, the distributed global knowledge is constructed by enacting practical interaction between modellers and by means of the network connection between their computer models. On the basis of distributed cognition, the knowledge is socially distributed across modellers. More importantly, this constructive process also describes and constitutes the social reality of agents by making their interaction observable and accountable. In this sense, this proposed framework, integrating cognitive and social processes, enables several modellers in a distributed environment to shape the agency of agents within an application.

4.3 Agency in AI, EM and DEM

Agent and *agency* are amongst the most important concepts in EM. As explained in Chapter 2, an agent can be an observable or a family of observables responsible for a particular state change. This definition is very different from the general usage of this term 'agent' in computer science, particularly in the Artificial Intelligence (AI) field. When several modellers are involved in a distributed environment, as in a real-life society, the definitions of agent and agency in EM furnish more practical and commonsense meanings.

The term 'agent' in computer science, particular in the AI field, generally refers to an entity which is granted or ascribed human-like mental qualities and is capable of interacting with its external environment [DBP93a DBP93b, KJ98, Rao94, Sho93, WJ95]. J. McCarthy argued that it is legitimate to ascribe human-like mentalistic notions to a machine when such an ascription expresses the same information about the machine that it expresses about a person [McC78, quoted in [Sho90]]. Similarly, D. Dennett has coined the term 'intentional system' to describe entities "whose behaviour can be predicted by the method of attributing belief, desires and rational acumen" [Den87, p.49]. These arguments underlie the fundamental concepts of agent and agency in the AI field.

As Wooldridge and Jennings discussed in [WJ95], descriptions of AI agents involve two kinds of notion: *action-based* and *mentalistic*. Action-based notions, such as autonomy, social ability, reactivity, and pro-activeness, refer to the ability of an agent to take actions. Mentalistic notions, such as goals, beliefs, plans, intentions, capabilities, knowledge, desires, and choices, refer to human-like concepts concerned with mental processes. Both kinds of notion are considered as the most critical factors affecting an agent's actions. Different agent models provide characterisations in terms of different notions. For example, Rao regards plans, beliefs, goals and intentions together as the mental states of an agent [Rao94], but Shoham takes beliefs, capabilities, choices and commitments into account [Sho93].

Although different mentalistic notions may be applied to an agent, it is generally agreed that given specifications prescribing their own notions directly control what an agent can do. In other words, these kinds of agents, here called AI-agents, are restrained from acting or causing effects within an unexpected context in order to make it possible to describe, explain and predict their behaviour in formal terms. From this standpoint, agency is interpreted as the capability attributed to an agent as specified by prescriptive mechanisms [WJ95, LdI95], and is granted by designers in ways that separate it from its context. Such agency is expressed through activity that, like reasoning in formal logic or a search for anticipated facts, is determined by a context that, though yet to be specified completely, is of a preconceived type.

An emphasis on knowledge prescription that does not take sufficient account of context is not without its problems. M. Minsky remarked that "formal logic is a technical tool for discussing either *everything that can be deduced from some data* or *whether a certain consequence can be so deduced*; it cannot discuss at all what *ought* to be deduced under ordinary circumstances." [Min74, p. 141, original emphasis]. In [Dre79], H. Dreyfus argued that what ought to be deduced can never be represented in the form of elementary context-free features characterising any type of content and structure of knowledge needed for this deduction. He also claimed that anticipating facts about a subject's total knowledge is an infinite task which must be situated rather than prescribed in a way of presupposing a background of cultural practices and institutions.

Indeed, it is increasingly recognised that knowledge prescription (or representation) has run into serious trouble in seeking to develop an intelligent machine that can act as a human being capable of adapting to its environment [Agr95, Cla97, WF86]. The context-free feature simply indicates the flaw of having far too little adaptability to accommodate frequently changeable practices. In this respect, the agency of an AI agent is rigidly limited to predictable description and cannot easily be adapted to a changeable environment.

Because of this lack of adaptation, the definitions of the terms 'agent' and 'agency' from AI field are not suitable for the framework of DEM. To give a suitable definition for both terms in DEM and EM, it is appropriate to review the following literal, dictionary definitions [Oxf89, Web61]. An 'agent' is defined as:

- One who (or that which) acts or exerts power, as distinguished from the patient, as distinguished from the instrument.
- A means or instrument by which a guiding intelligence achieves a result.
- Science: Any natural force acting upon matter, any substance the presence of which products phenomena.
- Of persons: One who acts for another; one entrusted with the business of another; a substitute; a deputy; a factor.
- Of things: The material cause or instrumentality whereby effects are produced; but implying a rational employer or contriver.
- The part of the system that performs information preparation and exchange on behalf of another. Especially in the phrase 'intelligent agent' it implies some kind of automatic process ... to perform some collective task on behalf of one or more humans.

The term 'agency' is defined in the following ways:

- The faculty of an agent or of acting; active working or operation; action, activity.
- Working as a means to an end; instrumentality, intermediation.
- Action or instrumentality embodied or personified as concrete existence.
- The office or function of an agent or factor.

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. From this point of view, an entity is an agent because of its agency. For example, the corner of a table can be an agent when it hurts someone's fingers or is used to open a bottle of wine. In the similar manner, a book can also be an agent when it changes the mind of the readers, increases the author's reputation or is used to knock on a door. These examples show that the agency of an entity cannot be specified with reference to its intrinsic features, since it is so widely open and undetermined.

In this sense, that an entity becomes an agent is due to the attribution of particular activities or causing effects. It does not matter whether it is active or passive, or whether or not it is able to perform these attributed tasks autonomously. At the same time, the consequence of acting or causing effects need not be guaranteed, since it depends highly on the combined context established by the agent's own status and the prevailing situation in its environment. In other words, anything is eligible to be an agent if its potential agency is granted (cf. the concept of view 1 agent in [Bey97]).

From this viewpoint, agent and agency are associated with situated activities rather than activities prescribed by rigid algorithms. For an AI-agent, the agency representing its capabilities is determined in a prescribed manner. As previously explained, this circumscribed specification is not sufficiently adaptable to a rapidly changing environment. By contrast, an alternative definition emphasising the aforementioned situatedness of agents is given in EM: *an agent is an observable or a family of observables responsible for particular state changes*[14].

Given this definition, the boundary between an agent and an observable hinges on the responsibility of the former for particular state changes. The responsibility attributed by the external observer to an agent has no intrinsic relation to either the agent's capability or its external environment. Neither is it to be interpreted as predicting the behaviour of the agent and the system. Instead, it is seen to refer to the modeller's experience and expectation. As long as the responsibility for particular state changes is projected, every observable/family of observables is immediately viewed as an agent.

For example, the handle on my door is an observable. It becomes an agent as soon as I turn it down to open my door since it is now responsible for changing the door's state. It does not matter that the handle of the door which I turn down is incapable of

[14] Without specification, the term 'agent' is used in the thesis in this sense. In addition, state changes are meant to refer to a change in the values and/or definitions of observables.

bringing about this attributed state change with my assistance; nor does it matter what the door's state is, such as whether it is locked or unlocked. Only if this responsibility for state change disappears, such as if I stop turning, will the handle of my door revert to an observable.

At the same time, no certain outcome is ensured for the attributed agency, even though this agent is responsible for the attributed state change. An agent engages with its responsibility according to its internal status, as defined, for example, by its abilities and intentionality, and its external situation, as defined, for example, by other agents' status. Uncertainty within both makes the result to a certain extent unpredictable.

Agent and agency defined in this way are strongly associated with the context of the external observer and the observed real world. The attribution of responsibility for particular state changes is not persistent nor are its execution and consequence guaranteed. In fact, all of these conditions can be altered at any moment, that is, they are situated and unpredictable. The 'protocol' part in the LSD account is used to convey this situated concept. In addition, an agent can be viewed as privileged to instigate change to some observables and dependencies. The modeller in EM is called a 'superagent', since no other modeller is privileged to change *all* observables and dependencies.

Different principles are used in shaping agency in EM and DEM. The shaping of agency in EM is accomplished by the interaction of the modeller with the computer model and ideally the referent. As described in section 2.2, the modeller, as the superagent, is able to change all observables and dependencies on behalf of other agents in a computer model. Each agent reliably performs the actions specified by the superagent (either explicitly or by automatic procedures) irrespective of its autonomy[15]. By means of repeated observation and experimentation, the agency of agents can be gradually shaped

[15] The supporting tool tkeden does not take into account each agent's context, including its own status and its environment situation. This account of reliable agents is acceptable in a stand-alone modelling environment because the context of each agent in this case is prescribed only by the super-agent.

in an open-ended fashion. This approach to shaping agency leads to a model that reflects the modeller's subjective, private and provisional perspective, and is an effective way to construct a computer-based model in EM that captures the modeller's construal [ABCY94]. It is also suited to the development of a computer model whose complex interactions are based on an aggregation of competing primitive stimulus-response mechanisms, resembling M. Minsky's model of the human mind as "a society of ever-smaller agents that are themselves mindless" [Min88].

Few agent models in AI take this situatedness of agents into account. Most conceive an agent as entity with innate properties, such as autonomy, reactivity, and social ability that are typically fixed and persistent. This association of properties with an entity takes no explicit account of the situated interactions between the agent and its external environment. Hence, the agency of an agent is determined by a set of essentially invariant properties of the agent that does not take account of its situatedness.

An exception to this norm is the agent model proposed by M. Luck and M. d'Inverno in [LdI95]. Within their model, an agent is instantiated from, and in turn reverts to, an object at some points in time. In contrast to the tradition of viewing agents as specialised objects, they argue that agency is transient for serving given goals. Without serving these goals, an object cannot be an agent. To some extent, this concept is similar to the situatedness of agents proposed in this thesis, but its focus lies mainly on formalising the appropriate agency of an agent in order to act for the intended goals. In this way, agency is defined in advance according to the intended goals instead of being shaped by repeated observations and experimentations, as in EM. In addition, the situatedness of agents described here should not be confused with the 'situated agents' concept proposed by Rosenschein and Kaelbling in [RK95]. The latter means that agents interact with their environment in a situated way, that is, to respond appropriately to diverse situations, as identified by prescribed rules.

The concepts of agent and agency in EM are adopted with a slight extension in order to apply to a distributed framework of DEM. An agent within DEM is not simply a mindless component of the computer model, which is reliably responsible for the state change given by the external observer to reflect his/her understanding of the agent's agency. Instead, by means of the enaction of the A-modellers, each agent can interact with other agents as if it is in its customary context. Corresponding to its current situation, each agent is engaged in situated actions accountable to others. The actions in turn reflexively create the reality of the interaction between agents. In other words, the agency of each agent, that is, the responsibility for state changes, is shaped in the process of creating the reality of agents' interaction rather than only in the individual cognitive process of the external observer in EM.

The shaping of agency has to be achieved by performing a cognitive process and a social process in parallel. After building up the individual computer models to correspond to the referent in the real world, as in EM, A-modellers as external observers shape the agency of individual agents in a cognitive process of experiential learning through repeated observations and experiments. In this way, the agency of an agent is simply localised in the mind of an A-modeller who is modelling this agent. Since an agent is only a part of a system consisting of many interlocking, interacting and mutually dependent components, its agency should be associated with those of other agents. In other words, the agency must be made accountable to other agents for the purpose of integrating with other parts.

In order to reach this point, a social process to do with construing the interaction between agents and constructing a working understanding among them is invoked. Each A-modeller acts as an agent by doing what the agent does in its practices in order to give sense to other A-modellers acting as other agents. Also, it is expected by the A-modeller that other A-modellers can do whatever else is accountable in their enacted agents' context. In this being-participant-observer way (cf. [AA87]), A-modellers adapting to the observed practice and to the reaction of others are led to shape their agency in a reflexively constitutive fashion. At the same time, the S-modeller acting as the rapidly changing environment also becomes involved in the social process to intervene with the interaction between A-modellers from the global view. As described in the last section, all modellers can collaboratively interact with each other in order to contribute to the embodiment and enrichment of the shaped agency of agents. The collaborative interaction for agency can be achieved through the incorporation of the following three modes:

1. without any interaction with others, through an individual cognitive activity of the A-modeller self, in which the interaction with the local computer model is invoked in order to maintain the virtual correspondence between the model and the referent (see Section 2.2);
2. through interaction with other A-modellers via the connection of all computer models, in which A-modellers act as agents in their (agents') ordinary context in order to reflexively constitute the agency of agents (see Section 4.2);
3. through the intervention of the S-modeller via a network connection with the computer models of agents, in which the new context triggers adaptive responses in the interaction between A-modellers (see Section 4.2).

As a result, the agency of agents is not only in the mind of the S-modeller, but is also socially distributed across A-modellers, as described above.

In practice, the agency of agents is reflexively constituted by a large number of local interactions and adjustments in practice. Many of these adjustments appear to reflect the common insight distributed across A-modellers emerging from two processes, that is, the social process and the cognitive process. To the extent that the exploration of agency adapting to a changing environment counts as learning, it may be said that shaping agency in DEM is an instance of collaborative learning that incorporates the experiential learning highlighted in EM.

4.4 Design and Evolution for SSD

There is an important and interesting difference between the processes of shaping agency in AI, EM and DEM discussed in the previous section. The first two attempt to shape agency through supervisory intention and intervention granted to the external observer, but DEM shapes agency through the process of evolutionary interaction between modellers for adaptation. This difference strongly resembles the difference between design and evolution for software system development (SSD). This merits further discussion here.

Design[16] for software development is an intelligence-intensive effort to build a software system and make it work. Its purpose is to create a software system whose behaviour is consistent with what the developer and ideally the user want to achieve. A key concern in this design activity is to predict the behaviour of the system to be developed under various scenarios of use. In this connection, the *predictive strategy* – in the sense introduced by D. Dennett in [Den87] – that is adopted by the developer is centrally important. In traditional SSD, system behaviour is typically predicted via an *intentional strategy* which "consists of treating the object whose behaviour you want to predict as a rational agent with beliefs and desires and other mental states exhibiting what Brentano and others call *intentionality*" [Den87, p. 15, original emphasis][17]. The concept of intentionality applies to all kinds of mental acts, for example believing, imagining,

[16] Here the general meaning of this term 'design' is used for the synthesis of conceptual design, physical design (or implementation) and testing.

[17]This predictive strategy is only made explicit in certain agent-oriented programming development approaches [Rao94, Sho90, WJ95], but is sufficiently general to account for most approaches to SSD. In particular, the intentional strategy can take account of both reactive components and user activity (cf. [Den87, p.17, p.20 & p.22]).

wanting, and so on [Hau97]. In the light of this discussion, traditional SSD can be viewed as an 'intentional strategy' for design, and the developed system as an 'intentional system' (cf. [Den87, p22-23]).

For more accurate and detailed prediction of the intentional system, the method of decomposition is exploited. By this method, other intentional strategies at a number of different levels of abstraction can be used in a descending series to decompose each higher-level strategy into a number of lower-level ones. After continuous decomposition, an intentional system can be developed by a collection of hierarchical intentional stances whereby the intentionality of the developer is then exhibited in the behaviour of the intentional system. In this way, the behaviour of an intentional system can be predicted, though the results may be flawed.

From the developer's perspective, the intentional strategy as a means for drawing up design specifications relies upon being able to conceive abstract entities capable of performing required functions. This is the appropriate way to organise the design activities before undertaking the detailed implementation in terms of formal abstracted notions, such as search trees, data structures, and evaluation algorithms. For example, in the object-oriented framework, the design specifications will be couched in terms of the 'objects', 'properties' and 'procedures', irrespective of whether or not these have been clarified in detail in the mind. The entities identified by the intentional strategy will be established by top-down analysis rather than by a bottom-up approach from the underlying mechanisms. This strategy presupposes an 'explanatory cascade' [Clar90], leading from top-level computational theory down to low-level elementary implementation, that needs to be taken on faith.

In the light of the above discussion, most design approaches proposed for software development, such as object-oriented design [Boo94, Jac92, CY90] and agent-oriented design [Sho93, Rao94], can be viewed as instances of implementing the intentional strategy. Within these approaches, the intentional strategy adopted by the developer is created from the clues that emerge from simplifying the object of study and postulating a set of rules for describing the behaviour of a system. Given this description in mathematical or logical forms, the developer can then prescribe the behaviour of the intentional system and refer its consequences to reality. If the consequences accord with reality, then the used intentional strategy is indeed feasible; if not, another intentional strategy should be implemented. Hence, the approach to design becomes a process of search optimised intentional strategy whereby the behaviour of a system can be predicted.

However, because of the growing size and complexity of today's information systems, descriptions specifying the system's behaviour (or intentionality) have become 'moving targets'. They are difficult to capture and pin down into a specification cabinet [BCDS93]. The need for incremental descriptions of such systems makes any preconception of the design system unreachable and inappropriate [Blu94a, Blu94b, Fis91, Bro87]. That is to say, it is in practice hard to describe clearly the top-level intentionality. As a consequence, it becomes problematic to adopt the intentional strategy for the development of complicated computer models having open or ill-defined requirements [Hen96].

Additional concerns are raised by design activities, such as verification and validation, that are used for the purpose of characterising an intentional system with the required predictability. These activities demand a rigid correspondence between the specifications and the implementation of the system's behaviour. The fixed relationship indicates the correctness of the developed system, but it also constrains the developed system from adapting to a rapidly changing environment. As a result, some inevitable problems emerge, when the developed system is applied to an open real world [Bro87, Bro95].

Today, it is often no longer acceptable if a developed system is correct but solves only the problems for which it was originally designed. Ideally, it should be able to grow and change in order to solve further relevant new user problems that emerge over time. In other words, 'building the thing right' and 'building the right thing' is not enough: today's systems have to be adaptable to the changing real world and cope with the diverse situated needs of users in solving their problems [FC96, Flo87, Leh98, LR98]. This fact highlights the inadequacy of an SSD based on the intentional strategy with step-by-step refinement in phase-based process models.

In fact, software design can be viewed as a problem-solving activity, and as such it is very much a matter of trial and error [Vli93]. I. Sommerville has provided the following helpful description:

> "[Software] design is a creative process which requires experience and some flair on the part of the designer. Design must be practised and learnt by experience and study of existing systems. It cannot be learned from a book. Good design is the key to effective engineering but it is not possible to formalise the design process in any engineering discipline." [Som92, p. 172]

In other words, the creative learning process of software design has to be unfolded by means of successive events of trial and error. Certainly, phase-based models provide only limited help. Neither the prescribed specifications for the designing product nor rigid engineering norms postulating the developer's activities can make the process progress in an effective and efficient fashion. For these models, trial and error is only possible in the form of retrospective feedback, that is, by learning and creating after the product has been produced. As a result, this afterthought often involves a very large additional and unanticipated cost. This consequence is evident in the fact that the maintenance cost of software systems is often very high [Sch90, Som95].

To incorporate trial and error in the intentional strategy, some models invoke the *experimental strategy*, in which one can treat elements of a system as rational agents and explore their behaviour through experiments. With the integration of the two strategies, these models, such as prototyping [Rei92, Mar91, And94] and scenario-based analysis [DF98, RSB98, SDV96, WPJH98], seek to explore and clarify the behaviour of the developing system through continuous experiments. The experimental exploration can provide the developer with experiential information on how the system will behave.

The two strategies are complementary for shaping the behaviour of the developed system. The intentional strategy is engaged in a top-down approach. It decomposes a higher-level strategy into a number of more accurate and detailed strategies by providing a less abstract level of descriptions for their design. On the other hand, the experimental strategy can be treated as a bottom-up approach in which certain parts of the system are clarified individually and incrementally integrated together. By means of both strategies, the developer can search for the optimal solution to prescribe the behaviour of the system.

In this respect, the behaviour of such a developed system (or an intentional system, in Dennett's term [Den87]) exhibits the developer's individual intentionality but also reflects the connection between his/her experimental experience and the system. Nevertheless, this also indicates that the development process and its products are characterised by the individual mental states of this external observer, including knowledge, understanding, judgement, and so on. The developed system is in effect closely related to the developer's individual cognition of the user's needs. As a result, it is very difficult for those models that involve both the intentional strategy and the experimental strategy (and these models that involve only the intentional strategy) for SSD to avoid the emergence of a gap between the developer's system and the user's system. As described in Section 1.1, this gap, that mainly arises from the cognitive difference between the developer and the user, has led to practical difficulties in SSD and in the use of the developed system in the user's environment. In order to narrow the gap, a strategy for evolution has been to some extent embedded in most models.

Evolution is a classic concern of SSD. Its feature of incremental refinement is found in many models for SSD. This feature can be embedded in the whole development process [Boe88, Yeh90], or just parts of it, for example, the implementation phase [Boo94]. B. Boehm proposed a spiral model to highlight the iterative framework of software development in which software systems are incrementally refined and enhanced. R. Yeh recommended a software evolution paradigm in which – in order to keep pace with the changing environment – validation and evolution are embedded in the whole development process, rather than performed as an afterthought upon completion of the development [Yeh90]. Instead of concentrating on the evolution of a software system in its development process, M. Lehman placed the emphasis on the evolution in its operational domain. He argued that the organisation of developing and maintaining a large software system should itself be regarded as a self-stabilising feedback system [Leh94b, Leh97, LR98]. The feedback from the operational domain (of users) is the main resource for the development of a software system, whereby the understanding of the developer and the structure of the software system are evolved.

Evolution in SSD can be unfolded in two stages. One is at the stage of system construction where developers are developing the system in order to satisfy the user's requirements. The other is the stage of system operation where the system is used in the user's actual practice. During the stage of system construction, evolution is meant to develop a software system incrementally in order to satisfy the emerging requirements of the user's intended system [Boe88, Boo94]. Evolution in the system operation stage is to refine the developed system in order to be consistent with the user's actual system [Leh94b]. Though they have different subjects, both types of evolution are processes where the developer's system, which is developed or still under development, is changed in order to coincide with the user's system, whether it is in actual use or is just being conceived for use. In short, evolution within the above models is viewed as a process in which the developer's system is evolved in order to narrow the gap between the developer's system and the user's system.

Hence, design activities for refining or enhancing the developer's system are applied to the process in order to bring the developer's system and the user's system as close as possible. However, this has not proved very satisfactory and successful. This is partly because the developer's system usually changes too slowly to keep up with the change of the user's system. A more important reason is that the evolution through the refinement and enhancement of the developer's system cannot rule out the emergence of cognitive differences between the developer and the user. As described in Section 1.1, being an 'outsider' of the user's system that resides in a rapidly changing environment, the developer can hardly reach a full understanding of the system. The developer's understanding of the user's system is usually limited and fragmented. It is not surprising that evolution invoked by the developer with his/her incomplete understanding still exposes the gap between both systems. For this reason, it is possible to regard the developer-centred evolution within these models as a form of design associated with the intentional and/or experimental strategies mentioned above.

Where shaping the agency of agents is concerned, it is plausible to regard EM as involving both the intentional strategy and the experimental strategy in an open-ended situated manner. The intentional strategy is embedded in the agency of certain agents whose responsibility for state change has been confidently described. And the experimental strategy, which can provide the modeller with practical experience in a particular situation, is invoked by means of 'what if' experiments in order to explore the agency of agents. With situated modelling (see Section 3.2), the implementation of the two strategies for SSD in EM becomes more effective and powerful for shaping the agency of agents. The intentionality of the modeller for shaping the agency is thus exhibited in the process of experimental exploration, and at the same time his/her empirical experience of how the system will behave in a particular situation is gained. In this respect, like those models that involve the two strategies for SSD, EM also indicates a process of individual cognition that is associated with the modeller's intentionality and empirical experience. This egocentric perspective of the modeller (that is, 'agency is in the mind of the external observer' [BeyMsc]) accords with Y. Shoham's assertion that 'agenthood is in the mind of the programmer' [Sho93].

A weaker notion of intentionality is taken into account in EM. The behaviour of the system developed by EM is not guaranteed to be permanent in the real world, though it still is expressed in the form of intentionality of the developer. Until confidence in the elements' behaviour has been established, theoretically the modeller can continue to perform 'what if' experiments to search for the optimal description of the agents' agency. This search process through both intentional and experimental strategies is consonant with D. Sonnenwald's concern for the knowledge exploration process for artefact design in which diverse knowledge from multiple domains, disciplines and contexts among specialists can be explored and integrated [Son96].

More significantly, DEM, that is proposed for clarifying and enhancing the distributed perspective on EM, involves a continuous process in which the agency of agents is evolved through their actors' (that is, A-modellers') interaction with each other in their (agents') ordinary context. This process is very similar to the evolution in an ecosystem for adapting to a changing environment by means of its natural emergent, situated activity [Hen96, AL89, SS95]. The evolution in an ecosystem, whose activities invoked for change (and their results) cannot be preconceived, is conducted by elements of the system itself rather than by an 'outsider'. It is not an exercise in retrospective patching by means of feedback with the aim of refining or enhancing the previous system.

Instead, it emerges from continuous local self-adjustments of the system in response to diverse situations encountered in a changing environment. Each local self-adjustment, by carrying out an incremental change to the system, may cause adaptive responses in other parts of the system. As a result, more local self-adjustments are invoked until each element of the system is stable in relation to the current environment. In other words, evolution within an ecosystem itself is carried out through the system's elements in interaction, and the details of its exploration process significantly affect the outcome. This evolutionary interaction between elements of an ecosystem and its changing environment enables the system to survive in the rapidly changing environment. Hence, evolutionary superiority results from just letting the system run by itself. In this case, there is no gap between the developing system and the actual running system because both are the same system.

In this respect, DEM involves the *evolutionary* strategy in which a software system is treated as an evolving system whose behaviour is adapted to its rapidly changing environment in a situated manner. Unlike the other two strategies mentioned above, the evolutionary strategy in DEM emphasises the role of interaction for adaptation. The interaction that is embedded in the evolutionary strategy leads the system to evolve so as to fit well into the current situation. It is unpredictable but powerful, because no prescribed algorithm is provided to constrain the evolution. The behaviour of the system is no longer persistent, since it could be adapted to a new situation at any moment. Like an ecosystem, the system evolves in response to its surrounding situation, and its evolution never ceases until it is thrown away. The evolutionary strategy proposed here for adapting a software system to a rapidly changing real world in an interactive manner is in line with P. Agre's view of an intelligent system: the structure of the system's behaviour is located in the interaction between the system and its environment [Agr95]. Note that evolution within DEM is accomplished by combining the evolutionary strategy with the intentional strategy and the experimental strategy. It is distinguished from design involving only the latter two strategies.

Certainly, it is very difficult for most traditional design models to support the evolutionary interaction between elements in a situated manner. Design within these models can be regarded as an exploratory search rather than an evolutionary process. This is because the behaviour of a system is designed by what J. Goguen has called 'introspection' where the elements and the causal links, or mechanisms, of the system are classified and identified through the imagination of the developer [Gog97]. Even given the experimental strategy mentioned above, the system behaviour is still bounded by certain prescribed descriptions. In other words, design is a searching process where the developer optimises the imagined description by continually changing the structure of mechanisms and the hierarchy of elements. This optimisation, making the system's behaviour quite describable and predictable, can presumably be achieved by providing enough consecutive events of trial and error. In this sense, these models are capable of supporting search but not evolution [AL89].

Within the framework of DEM, however, the evolutionary strategy can be employed in a natural fashion. It can be exploited in the two stages of evolution in SSD: system construction and system operation. During system construction, DEM empowers A-modellers to interact with their own computer models at any moment in order to adapt to a new situation happening in their observed worlds and/or their computer models. This new situation may immediately give rise to further adaptive responses from other A-modellers if this new situation on their computer models does not correspond closely to their observed worlds. Consequently, a continuous process of interactive adaptation emerges and at the same time the system evolves incrementally.

With the concepts of ethnomethodology, A-modellers in DEM interact with one another, their changing environment, and their computer models by pretending to be agents. In pretend play, it is as if A-modellers are affiliated to a society of agents and make their interaction with each other accountable. In this sense, it is plausible to view each A-modeller as an 'insider' of the system, who is capable of enacting the interaction between agents. Each interaction leads to a local self-adjustment on the A-modeller's computer model, and also brings the whole system and some computer models to a new situation due to the propagation of dependency.

In other words, A-modellers on behalf of agents invoke the evolutionary interaction for adapting the developing system to the current situation. This evolutionary interaction eventually leads to the evolution of the system. The concept of evolutionary interaction has been applied to the case study of a railway accident illustrated in Chapter 6 and also to develop a framework for modelling the situated process of requirements engineering discussed in Chapter 7.

The evolutionary interaction for A-modellers is situated. No result can be predicted because of the nature of situatedness and the uncertainty of the interactions that will be triggered [LR98]. As happens in an ecosystem, this interactive process should enable the system to converge to a stable state if the invoked interaction corresponding to the situated change is accountable to others and is adapted to the current environment. If there is a failure to reach convergence, for example, through conflict or inconsistency in the definition of an observable between A-modellers acting as agents, the developing system is unlikely to succeed. In other words, the system is evolved through the interaction between the 'insiders' of the system for adaptation in a situated manner. This evolution is very different from the developer-centred evolution mentioned above. The latter is carried on by the 'outsiders' of the system through design activities.

In a similar manner, the evolutionary strategy has also extensive application to the stage of system operation. Traditional design approaches aim at questing for a perfect software product without the gap mentioned above, but it is still like the Tower of Babel in the real world. This is because cognitive difference is inevitable while the developer, an 'outsider' of the developing system, develops the software system. From the perspective of users, the system they need is one that is able to solve their practical problems in time. In this respect, an adaptable system that can cope with emerging change may be more suitable for users in their real world than a perfect system. It is because invoking developer-centred evolution as suggested in traditional models is too late and expensive, due to the problem of tacit knowledge [Gou94]. Moreover, it is true that no one can do the adaptation better and faster than users, if they are qualified enough, since only they know what is needed at this moment for solving their current problem. In fact, the concept of enabling users to adapt their systems to a rapidly changing environment has been well implemented in a spreadsheet system. Users of a spreadsheet system initially develop a spreadsheet, and then adapt it by themselves to their changing practices. In other words, spreadsheet use is evolution of a spreadsheet by the users themselves. In this sense, the use of spreadsheets blurs the distinction between users and developers.

Focusing on users, rather than relying on the afterthought activities of an outsider for refining or enhancing the developed system, is an alternative way of bridging the gap between the developed system and the actual system. This means letting users adapt the system in use by themselves. To achieve this, it will be necessary to enhance the user's ability to perform 'end-user programming'. The developed system must also be characterised by considerable adaptability, so as to enable the user to evolve the developed system in a timely fashion. The first step towards enabling evolutionary interaction to occur at the operation stage is to create a system that is readily adaptable to its rapidly changing environment.

However, most design approaches end up with a final product that provides little support for adaptation in a situated manner. Within this final product, all knowledge associated with the development of the system, such as its architecture, data structure and functionality, must be determined and unchangeable before the product is operated in the user's environment. This is because the knowledge embedded in the final product is fixed and determined by optimisation aimed at satisfying a prescribed goal. The frozen boundary of the final product reduces the risk of its being an incorrect or incomplete system but also prevents the developed system from dynamic modification for adapting to the user's practice.

In addition, in many cases, users are not allowed to modify the final product, even if they are sufficiently qualified and the modification is urgently required. This is because, from the developer's viewpoint, any change, no matter whether it is for correction, perfection or adaptation [Fai96], should be finished by taking the whole system, rather than local parts, into account. Any local change can affect other parts of the system and then lead the system into a mess if this affect is not taken into global account. It is recognised that local changes normally introduce more errors and lead to higher costs [STM95, Som95]. Indeed, this demand for a whole-system perspective is understandable and inevitably necessary in conventional top-down approaches for SSD.

Therefore, it is clear that the crucial issue of invoking evolutionary interaction during system operation is the difficulty of building a system that is adaptable through local change. In the light of the definitive programming described earlier (Section 2.3), the need of propagating local adjustment to global change is met through dependency maintenance. Any change in a local part can be propagated to the whole system and affect the other elements if the change is authorised. In other words, the user can make a local change for satisfying his/her individual need and/or for adapting to the current situation without taking global propagation into account.

With the benefit of definitive programming, evolutionary interaction can be accomplished through local changes from the users of the system. That is, the system can be evolved through its qualified users in interaction during the stage of system operation. Clearly, the capability of users (such as programming skill) is another critical factor for the success of the evolutionary interaction in the stage of system operation. Hopefully, performing local change to the evolutionary interaction is usually less difficult and demands less sophisticated skill in programming. Also, advances in end-user computing have increasingly reduced the problems of incapability [DL91]. In other words, in the near future, the capability of users should not be a big problem for invoking evolutionary interaction. The trend towards enabling the user to adapt the system to diverse situations in use accords with the demand for *design-in-use* in which the design activities are extended to the operational domain of users in order to meet their changing needs [Fis93, SH96].

By supporting the evolutionary strategy proposed here, DEM can bridge the gap between the developed system and the user's actual system in a significant way. In order to make the developing system as close as possible to the intended system, A-modellers can act as agents as if inside the intended system during the system construction stage. Recognising the fact that the user's need is never complete, the goal of the evolutionary strategy is to build an adaptable system that can evolve in a situated manner. Given such a system, the user, on the basis of his/her individual profession and practical experience, can adapt the system to his/her rapidly changing environment. Accordingly, the adapted system can always keep up with the user's actual system. In fact, the system-under-development converges to the system-in-use.