Chapter 6
Case studies: practising Empirical Modelling in collaborative contexts

In the previous chapters, I have shown that Empirical Modelling (EM), based on a different philosophical foundation from mainstream computing, potentially facilitates collaborative modelling particularly in respect of the three highest degrees of engagement (i.e. collaboration, co-evolution, and co-construction). I have also shown that the EM conceptual frameworks and the associated tool support that were conceived for concurrent engineering and distributed organizational work may not be expressive enough for the diverse and typically ill-structured activities that arise throughout the collaborative modelling process, e.g. groupware development. In these cases, it is preferable for the collaborative environment to allow modellers to interact using diverse modes of communication and to shift their roles seamlessly. While a full-fledged ideal collaborative environment is still under development, we still see benefits in studying the efficacy of EM for collaborative modelling in practice[90].

In this chapter, I discuss two case studies in software development projects where the core activity can be viewed as a collaborative modelling process throughout the development lifecycle, and two case studies in collaborative modelling that exploit an enhanced variant of the distributed EM environment developed by the author. The case studies in software development are designed to study collaborative modelling at the project level when practising an EM approach, while the case studies that exploit the enhanced variant of the collaborative EM environment are designed to study collaborative modelling at the

[90] Despite the imperfections in the tool support, EM may offer a better theoretical foundation for collaborative modelling.

interaction level.

In section 6.1, I present a case study on the Virtual Electronic Laboratory (VEL) project. This was a joint project between two MSc students (D'Ornellas, 1998; Sheth, 1998). Although the common goal for the students was to develop software to teach elementary electronics in a classroom environment, they had different perspectives in developing the VEL model. D'Ornellas focused on the simulation of electronics and showed the openness of the EM approach with an illustrative example in integrating the traffic light model (EMPA: trafficlightMendis1997) with the electronic simulation module of the VEL model. Sheth focused on a complementary objective: developing a multi virtual-agent interface and exemplifying the four interaction styles associated with the DEM framework in a classroom-teaching context. Unlike previous research by Maad (2002), this case study focuses on the co-construction of the VEL model and the discussion of the particular scenario adopted in relation to other possible scenarios that can be applied in a collaborative modelling approach.

In section 6.2, I discuss a case study on two collaborative modelling sessions of a distributed Jugs (d-Jugs) model, which is similar to the Jugs Model (EMPA: jugsBeynon1993) in the EM project archive. The purpose of this case study is to explore the interactions among a small group of modellers in ad-hoc synchronous collaboration[91] when practising an EM approach to collaborative modelling. This case study consists of two d-Jugs modelling sessions. Multiple data collection techniques were used in both modelling sessions. These techniques included participant observation (Spradley, 1980), video recording, screen video recording, and interaction history logging. The planning within the modelling process was kept to the minimum – neither of the modellers was briefed before the modelling sessions and the development time was very limited. This configuration gave rise to role-shifting behaviour of the modellers and revealed the dialogical character of the interactions involving script exchange to the observer of the modelling sessions.

In section 6.3, I discuss a case study of a collaborative modelling session of a Sudoku

[91] I also call ad-hoc synchronous collaboration "real-time" collaboration.

model. As in the case study of d-Jugs modelling, multiple data collection techniques were used in the case study of the collaborative Sudoku modelling. These techniques included participant observation (Spradley, 1980), video recording, and interaction history logging. The room configuration for the collaborative modelling session was similar to that in (Flor and Hutchins, 1992), where the modellers were sitting side-by-side and working on a shared model through separated but networked workstations. During the collaborative modelling process, the modellers were asked to develop a strategy to tackle a Sudoku puzzle collaboratively. They were encouraged to customise their collaborative environment when necessary. Despite the fact that the modellers were 'jump-started' with an existing model (EMPA: sudokuKing2005), they spent much of their time learning about the problem through interacting with the Sudoku model during the collaborative modelling session, due to their unfamiliarity with the problem context. Coincidentally, this made the modelling process similar to the early stage of traditional systems development, where developers are spending much of their time in learning the problem domain. This case study has exposed the diversity of human interaction and the efficacy of instant feedback[92] in synchronous collaborative modelling when using an EM approach.

In section 6.4, I study an undergraduate software development project on cricket simulation (Beynon, 1993). During the early stage of the software development process, the students were asked to practise an EM approach instead of a traditional approach to requirement elicitation, analysis and design. The aim of this case study is to reveal the potential benefits that EM may offer to collaborative modelling – where software development can be conceived as a specialization of collaborative modelling.

[92] Instant feedback is a feature provided by the principal EM tool (tkeden) and its variants. This includes state changes due to triggers, dependencies, and propagation of script changes from peers.

	Case Studies of Software Development Projects		Case studies that exploited the enhanced variant of the collaborative EM environment	
	Cricket (§6.4)	**VEL (§6.1)**	**d-Jugs (§6.2)**	**Sudoku (§6.3)**
Intended focus of the case study				
Project level	x	x		
Interaction level		x	x	x
Structure of the collaborative modelling process				
Structured		x		
Semi-structured	x		x	
Unstructured				x
Length of the collaborative modelling process				
Between 1 to 3 hours			x	x
Between 2 to 6 months	x	x		
Temporal character of the collaborative modelling process				
Asynchronous collaborations	x	x		
Synchronous collaborations			x	x
Spatial character of the collaborative modelling process				
Remote collaborations[93]	x	x		
Collocated collaborations			x	x

Table 6.1 – Considerations that the case studies have addressed

As table 6.1 depicts, the selection of case studies covered a range of frequently occurring situations and configurations that may be involved in collaborative modelling (cf. §5.1). The variety of factors considered in this selection include:

i) *Group size* – a range of number of collaborators from a pair of modellers to a medium size software development team with about a dozen developers.

ii) *Length of the collaboration* – a variation of the length of the collaborative modelling process ranging from a few hours to a few months.

[93] In both the cricket project and the VEL project, there is no 'firm' evidence to indicate that either remote or collocated collaboration was the only kind of collaboration. However, despite the fact that the modellers might have collaborated while collocated, I would argue that their collaboration was primarily remote. This is because:

1. At the time when the cricket project was carried out, there was no distributed (and collaborative) EM tool support available. Certainly, it is possible that the modellers discussed their models around a shared screen. Indeed, there were face-to-face project meetings, and collocated interaction between the modellers outside the cricket project was inevitable, as they were all enrolled in computing-related degree programmes at the same University. However, it is more likely that modellers were primarily working alone due to the nature of the coursework (despite the fact that the cricket project was a group project, one of their preliminary tasks was to submit an individual cricket model, cf. §6.4).
2. In the VEL project, the division of labour between the modeller pair was so clean that the modellers could work separately in isolated modelling spaces until late in the collaboration process, when integration of their individual models took place. The project work was planned in such a way that there was no strong need for collocated collaborations.

iii) *Structure of the collaboration* – different levels of structure in respect of the common goal.

iv) *Temporal character* – synchronous and asynchronous collaborations.

v) *Spatial character* – collocated and remote collaborations.

The motivation behind these case studies is:

i) To examine the effectiveness of the instant feedback provided by EM tools which potentially facilitates collaboration;

ii) To show that the interaction between modellers is similar to a dialogue, and has to be understood with reference to how their communication is enabled by their prior experience[94], as they negotiate the meaning of the evolving artifact through exchanging scripts;

iii) To explore the role-shifting behaviour of the modellers in synchronous collaborative modelling and to examine whether the conflation of the context of construction and use facilitates seamless transition of roles.

iv) To reveal the diverse interaction styles during the modelling process (in synchronous, semi-synchronous, and asynchronous collaborations) which stem from the dynamic nature of group work;

v) To assess the significance of tool support[95] in collaborative modelling and in particular whether or not we need tool support. If yes, what is needed in the tool support, and how can that be supported?

Through these case studies, we understand how EM principles may be applied in practice. Indeed, these case studies provide evidence of the potential of an EM approach to collaborative modelling. In the case of software development, the core activity can be viewed as a co-construction of an Interaction Situation Model (ISM). The idea of practising

[94] This includes domain knowledge, and both past and immediate experience in relation to the context of observation.

[95] The case studies reflect a paradigm shift in EM research, from traditional group work to interactive group work.

an EM approach to collaborative modelling has influenced the thesis author to develop a conceptual framework for participatory groupware development (see Chapter 7).

6.1 Case study I: the Virtual Electronic Laboratory project

There is evidence suggesting that the development and the use contexts are conflated when practising EM in a distributed systems development context. For instance, modellers in the Virtual Electronic Laboratory (VEL) project act as both the developer and the users (the role of teacher and student) at different times in the EM process. Previous accounts of the VEL project focused on its pedagogic potential in the context of collaborative learning (e.g. Beynon and Maad, 2002, Maad, 2002). In this section, I shed light on the VEL project from a collaboration perspective. In particular, this case study focuses on the interaction between the modellers.

6.1.1 The background of the case study

The Virtual Electronic Laboratory (VEL) is a distributed EM model that was collaboratively constructed by two MSc students, D'Ornellas (1998) and Sheth (1998), in a joint project. The motivation of this project was to examine and demonstrate the potential of EM in supporting learning in a distributed environment, in particular, supporting the learning of elementary electronics through experimentation in a virtual laboratory environment (D'Ornellas, 1998; Sheth, 1998). The development of the VEL was inspired by several typical real-life electronics laboratory scenarios[96] (Sheth (1998)). The project, in its very essence, was

[96] Sheth (1998) described four scenarios that are commonly found in real life electronic laboratory: (i) After the teacher and the laboratory assistant set up the equipment, they continue to set up the chosen circuit, with all the components on a circuit board. The students crowd around the teacher's table whilst the teacher first describes the circuit and its components. Then, the students are expected to observe and make notes while the teacher is changing the values of the components during the experimentation. In this scenario, the teacher is in control of experiment. (ii) Students are experimenting either in groups or individually. The students may either be given a pre-set circuit board with all the components on it, or be asked to build a given circuit on their own. In this case, the teacher does not have an overall control of the laboratory session: the teacher may walk around and assist individuals or groups one at a time. (iii) Similar to (i), but the teacher carries out the experiment with a computer tool instead. The advantage of computer simulation experiment is that the teacher can demonstrate drastic effects to the students, such as blowing the resistors up with an extreme voltage, which would normally be regarded as dangerous or wasteful when experimenting with real electronic components. (iv) Each student experiments with a computer tool, e.g. PSpice, on a private workstation. The students will be given the experimenting circuit and they

educational.

However, the virtue of the VEL project in the context of this thesis is not derived from its connection with computer-based learning, but its connection with collaborative modelling. At the time the study was carried out, the VEL was one of very few EM models in the EM archive (EMPA) that had fully exploited all four modes of interaction[97] that are closely associated with dtkeden, the tool for DEM:

i) In the broadcast mode, all the changes done at the teacher's workstation (i.e. s-modeller in DEM) within the modelling environment will be relayed to the students' workstations (i.e. a-modeller in DEM). The relayed components include the circuit and the model itself. The teacher and the laboratory assistant can prepare a circuit while the students are entering the laboratory. The teacher then explains the circuit and carries out the experiment in the same mode. Students watch the teacher's demonstration through their own workstations and make sense of it.

ii) After setting up the circuit in the broadcast mode, the teacher can choose to switch to the private mode. In this case, the teacher is letting the students carry out the experiment on their own.

iii) In the privileged mode, the teacher can assign access privileges on the components in the model to individual students or groups such that they can build or experiment a circuit cooperatively.

are responsible for 'drawing' the circuit within on their own workstation. Then, the students can start experimenting with the circuit, make observations, and learn from it. Similar to scenario (ii), the teacher may walk around the classroom and assist those who are struggling.

[97] It is worth noting that the four modes of interaction that are closely associated with the dtkeden and DEM are not exactly the same as those I described in chapter 5. For more details, please refer to section 5.1.3.

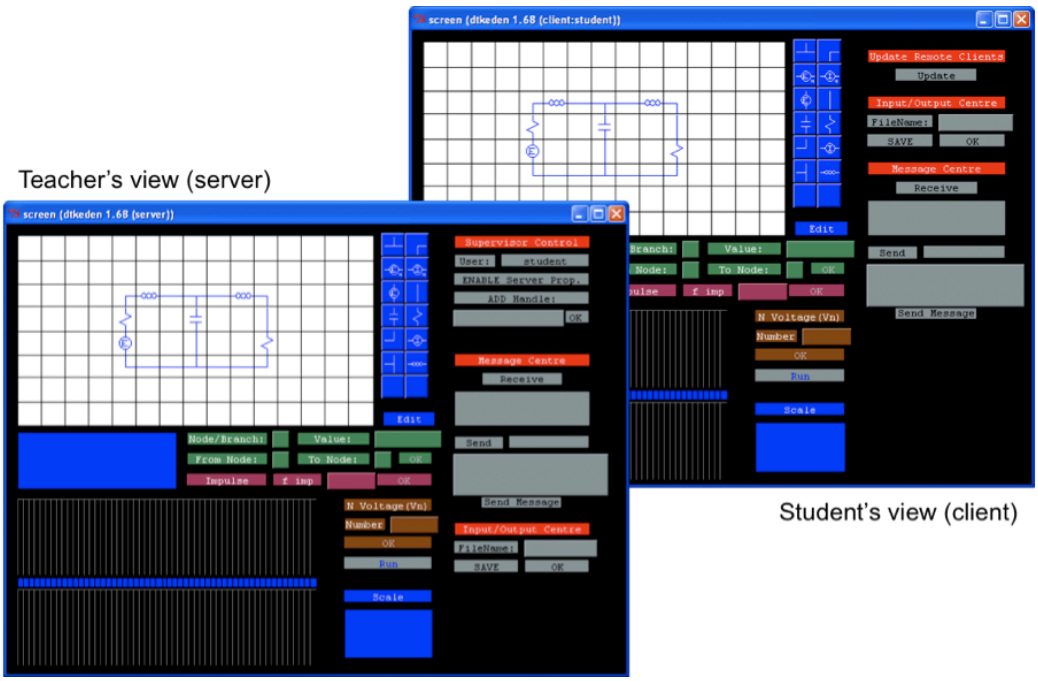

Figure 6.2 – A screen capture of the VEL model (EMPA: velShethDOrnellas1998) within the dtkeden modelling environment

iv) In the interference mode, the teacher selectively accepts redefinitions of the circuit components that are submitted by students. This can be used in, e.g. group discussion, where the group representatives (when students are working in groups) or individuals (when students are working alone) negotiate their contribution for integration into the public model.

As I will discuss later in section 6.1.5, the virtue of the VEL project is its use of the DEM framework for collaborative modelling. This in turn becomes a significant piece of evidence of the potential of EM for collaborative modelling at the cooperation degree of engagement.

Figure 6.2 shows a screen capture of the VEL model. Further discussion of the technical detail of the VEL model is beyond the scope of this study. For such details, the interested reader can refer to the MSc project reports written by D'Ornellas (1998) and Sheth (1998).

6.1.2 Method of study and approach to data analysis

This case study aims to reconstruct a description of the cooperation and coordination between the developers of the model. For this reason, the discussion is more appropriately regarded as an interpretive account that gives particular attention to the collaborative

aspects of the project. Due to the age of the project, studying the project is far from straightforward; not only was it impossible to contact the developers, but the incomplete interaction history in the definitive scripts also posed a challenge. Consequently, this case study is largely based on the dissertations written by the developers and the archived definitive scripts of the VEL model (EMPA: velShethDOrnellas1998). This also makes the data analysis within the VEL case study similar to document analysis as described in literature in research methods such as (Duffy, 2005) and (Yin, 2009). Duffy (2005) suggests that documents can be classified as *primary* or *secondary*, *deliberated* or *inadvertent* sources. Following Duffy's classification of documents, the VEL case study that is discussed here involves both primary and secondary, and both deliberated and inadvertent document sources:

i. Both the developers' dissertations and the archived definitive scripts are primary source, and the other case studies of VEL by other researchers (e.g. (Maad, 2002); (Beynon and Maad, 2002)) are secondary source.

ii. Developers' dissertations, on the one hand, can be classified as *deliberated sources* because the developers might have attempted to preserve some "facts" about the development process of the VEL project. Archived definitive scripts, on the other hand, can be classified *inadvertent sources* because they were produced as one of the 'deliverables' of the VEL project rather than as a preservation of the development process. [98]

However, the archived definitive scripts are more than merely texts. This is because they can be interpreted in the EM modelling environment (viz. tkeden or dtkeden). Due to the nature of EM (cf. chapter 4), the archived definitive scripts can, to some extent, also be used to recapture the *state-as-experienced* by the modellers in developing the VEL model.

Before approaching the data, I expected this case study to shed some light on, though not

[98] Duffy (2005) classifies primary document sources into two catagories, namely, *deliberated sources* and or *inadvertent sources*. He explains that *deliberated sources* "are produced for the attention of future researchers" and these sources "involve a deliberate attempt to preserve evidence for the future, possibly for purposes of self-vindication or reputation enhancement." (ibid, p.126) Duffy also explains that *inadvertent sources* "are used by the researcher for some purpose other than that for which they were originally intended." (ibid, p.126)

be entirely devoted to, some research questions that I had in mind. For instance:

i. How did the developers collaborate in relation to the aspects of collaborative modelling described in chapter 5, namely, the degree of coherence, the relationship between the modellers and the agents, and the modes of interaction?

ii. Have the developers followed the interaction styles proposed in the DEM framework?

iii. Does EM hinder or facilitate the development process?

iv. What characteristic(s) of EM, if any, has the project demonstrated?

Since no similar type of analysis of research into EM had been done before this case study, it was unclear how the developers' dissertations might reveal. Taking this into account, I adopted a flexible approach to the data analysis in the VEL case study: firstly, consult the developers' dissertations, then, based on their description of the collaborative modelling process, analyse the archived definition scripts.

In order to find out how the developers approached the VEL joint project, I analysed their dissertations and highlighted the pieces and sections that describe the process of development and the cooperation between them. The original plan was that the analysis of the developers' dissertations would give a basic idea of how the developers might have collaborated, and the challenges (in relation to collaboration) that they faced and overcame during the VEL project. This information would then be used to guide the analysis of the archived definition scripts. Unfortunately, the dissertations neither explicitly discuss the degree of collaboration involved in the joint project, nor give details about how the developers collaborate in the joint project. The developers' dissertations were focused on the educational aspect and technical aspect. Despite the fact that the project was a collaborative effort, there is little information about the collaboration process between the developers apart from the declaration sections, the acknowledgement sections, and the following observation:

> *"This would be a two-part project in corporation with another MSc colleague, Hansel D'Ornellas" (Sheth, 1998, p.5) "… this is a joint project and another objective is to allow the teacher to draw the circuit diagram and then extract the data and place it into the wanted matrices format for manipulation by my colleague. The extraction will be dealing with issues such as node labels, component values and so on. A part of the interface would be to display graphs about the circuit." (ibid, p.5)*

Furthermore, the developers' dissertations indicated that there was little or no overlapping between the their contributions in the VEL joint project:

> *"… my part of the project consisted of designing and implementing a user interface so that the teachers and students may collaborate with the tool. The output from the circuit analysis was to be done by my colleague. It was necessary to extract the data from the circuit model drawn, and to represent it in the required matrix format, ready for processing. This was achieved and my colleague was hence able to provide output from the matrices." (Sheth, 1998, p.52)*

To find out how they cooperated in more detail, that is, to determine how they divided the labour in respect of model construction, I have compared their contributions in the VEL project archive (EMPA: velShethDOrnellas1998). The analysis of the archived definitive scripts, to a large extent, involves a kind of comprehension that is similar to program comprehension (cf. e.g. Beynon and Sun 1998).

It is clear that two different styles of structuring the definitions and two different styles of file naming were used in the definitive script of the VEL model. For instance, one set of files were named a suffix of ".lib.e", while the other set of files were named without the use of any suffix. One set of scripts seems to be feature-oriented (i.e. one triggered action per script), and the other set seems to be agent-oriented (i.e. big scripts which consist of all relevant definitions of agents). This somehow echoes the claims in the developers' dissertations that the project was a cooperation between the two developers.

In order to know what division of labour was adopted by them, I have analysed the 'ownership' of the observables, dependencies, and agents in the scripts. In contrast to traditional computer programming, EM models offer an unusual resource for such analysis: dependency. As mentioned in chapter 4, dependency in EM is not the same as dependency injection or the concept of dependency in Enterprise Java Beans. Through tracing the dependency structure within the joint VEL model, it reveals how dependencies are used to link observables, agents, and triggered actions together – this provides us with an insight into how the developers might have collaborated. The inspiration behind this analysis is de Souza's (2004) work, where he suggests that the social dependencies among the developers are by and large captured in the artifacts they produce.

6.1.3 Limitations

The VEL case study is largely based on the analysis of archived documents. The major weakness of such an approach is that bias can easily be introduced either by the researcher in document selection (Yin, 2009) or by the document authors who may not declare implicit assumptions properly (Bell, 2005; Yin, 2009).

To some extent, these limitations have been addressed in the research process. For the first issue, i.e. document selection, this case study has included all the accessible documents that discuss the VEL project (cf. §6.1.1). While best effort has been made to report and analysis the VEL project as fairly as possible, this does not prevent unknown bias in the documents that was introduced by the documents' authors. For instance, the developers' dissertations might have reported their intended cooperation – but not their actual cooperation – in the development process. To verify whether the degree of the collaborative activity in the development process of the VEL model was genuinely limited to what I have classified as cooperation with a high level division of labour, I have also consulted the students' project supervisor, Meurig Beynon, informally. According to Beynon, it was likely that the developers had limited opportunity to work together. He explained that the developers were living in two different sites and were working on the project during vacation time. Furthermore, I have analysed the archived definitive scripts from the VEL project archive (EMPA: velShethDOrnellas1998). However, the analysis of the definitive scripts

does not tell the story by itself – it requires careful and skillful interpretation which involves knowledge of EM and of the EM process. This, in turn, is subject to possible bias in the interpretation. However, I would argue that any competent EM practitioner might arrive at similar, if not the same, conclusions if they analysed the documents in the way described in this case study.

6.1.4 Discussion

Initial coordination: division of labour

Due to the differences in their expertise, D'Ornellas and Sheth were observing from different perspectives and modelling from different contexts. On the one hand, D'Ornellas focused on the simulation of electronics underlying the graphical user interface due to his strong background in electronic engineering. On the other hand, Sheth focused on the graphical user interface (GUI) for both the teacher and the students. Although the two modellers were working as a team and modelling in separated spaces, there is no evidence to suggest that there are overlapping areas in their models – i.e. the coordination between the two was effective, if not perfect. Therefore, the next question in this case study is "How did they actually coordinate and cooperate?"

How did they actually coordinate and cooperate?

In traditional software development, developers often coordinate through interface components (deSouza et al., 2004), either well-defined in specifications or through mutual agreement between developers. In searching for the way that D'Ornellas and Sheth coordinated, I attempted to find the similar interface component in the VEL model. On this matter, Sheth (1998) provided a crucial hint on this "component":

> *"… this is a joint project and another objective is to allow the teacher to draw the circuit diagram and then extract the data and place it into the wanted matrices format for manipulation by my colleague." (ibid, p.5)*

The analysis of the definitive scripts of the VEL model reveals that D'Ornellas and Sheth were engaging in coordination. This was done through a two-dimensional matrix and semaphoric observables. The analysis also suggests that the two developers were

coordinating and cooperating, and not practising any higher degree of engagement (cf. §5.1.1) despite the fact that they were co-constructing the VEL model.

Unlike other programming paradigms (e.g. object-oriented programming), modelling with definitive scripts means that agents are connected together with dependencies (cf. §4.1). The EM model cooperatively constructed between D'Ornellas and Sheth also revealed how dependencies (also known as triggered actions in this case) were used to connect various key agents between the GUI mode (construed by Sheth) and the circuit simulator model (construed by D'Ornellas). The GUI model gathers the components' properties on the circuit board (cf. figure 6.3) and transforms them into a two-dimensional matrix ('A'). When matrix *A* is updated, the values of the semaphoric observables (*start_sim* and *ready_status*) will be updated by the *superAgent*. The update of *A* also triggers the computation of matrices *Yn* and *Jn*. After *Yn* and *Jn* are computed, this triggers the computation of *Vn*, and so on and so forth. After the last matrix is computed, the *dataCollect* agent is triggered, which updates the magnitude graph and the phase graph respectively. Figure 6.4 shows how the agents are linked together with triggered actions. The use of dependencies, in this cooperative context, not only simplified the definitive scripts a great deal[99], but also potentially made the cooperation easier through easing model comprehension – it allows the other developers to *see* the "structure of the model" through the dependencies[100]. In other words, the use of dependencies potentially allows modellers to "*see through walls*" to know what is happening at their peers (deSouza et al., 2004).

[99] This is because the definitive scripts do not contain an excessive number of listeners or other polling mechanisms that check for the latest data. This potential merit of dependency is also identified by Harfield (2008) in a case study in which he compared the modelling of Jugs using EM and using an object-oriented programming paradigm.

[100] The application of dependencies, in fact, is not restricted to facilitating collaboration; it can also be applied in other parts of the model (cf. §4.1). For instance, the underlying data structure of the circuit (cf. figure 6.1) is in a tabular form that is similar to that is used in spreadsheets. This potentially allows the use of dependencies between 'cells' where the components are located.

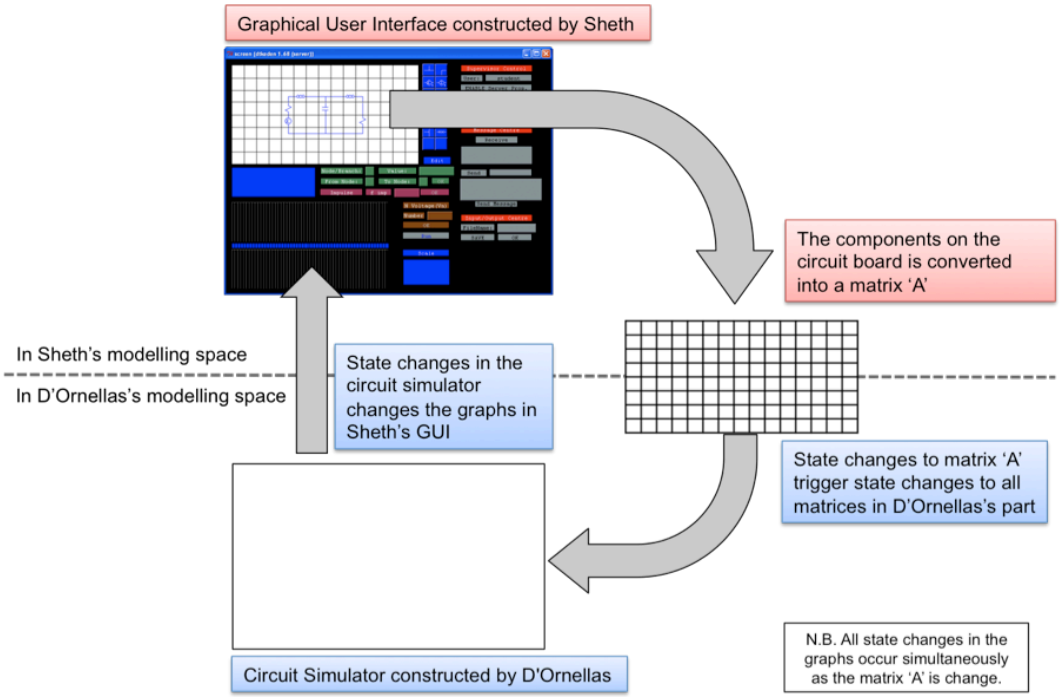

Figure 6.3 – Cooperation between Sheth and D'Ornellas in the VEL project

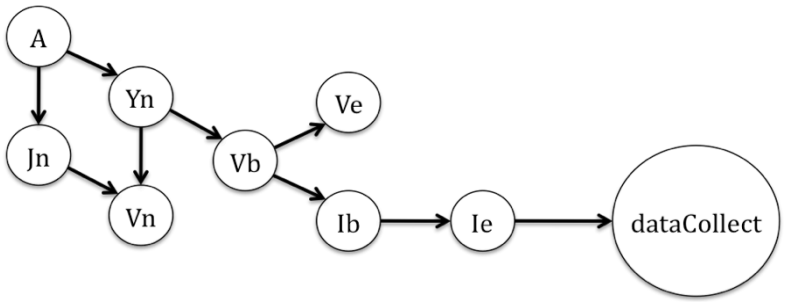

Figure 6.4 – Triggered actions graph after Matrix A is changed in the VEL model

Apart from revealing the cooperation, the analysis of the definitive scripts suggests that the two modellers may not have strictly followed the DEM framework during the modelling process – there are indications that they may have played the role of external observers, where in DEM, the subordinate modellers (or primitive modellers) are supposed to model agents through pretend play and internal observation (Sun, 1999). In the modelling process of the VEL model, there are only two modellers. However, there are more than two agents in both their individual models. It would not be possible to integrate all the agents together without external observation. This indicates that the modellers, at some point, must have observed the individual agents from an external perspective in order to investigate the interaction among those agents, either in the joint model, or in their individual models. Such

a modelling process does not follow the I-modelling as proposed and advocated by the DEM framework, but more likely resembles the MMMA scenario I described in section 5.1.2. This, in turn, resonates with the call for better conceptual framework to apply EM in collaborative modelling context.

What potential of EM has the development process of the VEL demonstrated?

DEM allows a developer to play the role of an agent and make observations from a perspective inside the agent (Sun, 1999). In relation to the *pretend play* characteristic of DEM, two agents in the VEL deserve some attention in this analysis. In the VEL project, the teacher and the student are viewed as agents internal to the "VEL system". From an EM perspective, the role of the teacher and the student agents are not prescribed[101] before the modelling process; they emerge and are shaped through continuous observation throughout the EM process (cf. §4.1). The adoption of the DEM framework allows two modellers to play the role of the teacher and the student as if they were in the use context of the VEL, and to model from these perspectives when they gain new insight stemming from the experimentation with electronic circuits. In this way, the model will evolve through the interplay between the use of the model and the redefinition of it, within one and the same modelling and experimentation environment. This potentially conflates the contexts of development and use, which leads to the conflation of the roles of developer and user, as I discussed at length in chapter 3.

Apart from the potential for coordination and cooperation, the VEL project also demonstrates the flexibility and openness characteristics of EM. As mentioned in section 4.1, EM models are subject to refinement through further investigation and further insights of the modeller. In the VEL model, not only can the components (e.g. circuits) be changed on the fly, so that the laboratory environment supports the experimentation with electronic circuits, the laboratory environment, as a part of the EM model, is also open to change. For instance, D'Ornellas (1998) illustrates how the VEL model be integrated with Mendis's traffic lights

[101] Strictly speaking, there is not even an initial presumption that the agents are to be played by humans, therefore 'users'. Only through the EM process does the role of these agents begin to be disclosed until it becomes clear how and why they should be played by humans, and therefore constitute a "context of use".

model (EMPA: trafficlightMendis1997) for learning how a square wave generator can be used to control traffic lights at road junctions. Such a metamorphosis of the EM models (cf. Beynon and Sun, 1999) through connecting two originally unrelated EM models cannot be effected without the insight and ingenuity that the EM modeller has established through the modelling process.

6.2 Case study II: the development of a distributed Jugs model[102]

In section 5.3, I described how EM might support different degrees of engagement in collaborative modelling. In the previous section, the case study of the VEL project revealed how modellers might coordinate and cooperate throughout the EM process. Although the VEL case study revealed a number of implications of EM, the collaboration during the modelling process was nevertheless asynchronous[103]. Furthermore, in respect of the degree of engagement in collaborative modelling (cf. §5.1.1 and 5.3), the VEL case study only revealed EM's potential for coordination and cooperation,. This motivates us to study how modellers collaborate and co-construct through EM in synchronous (i.e. real-time) collaboration through the case study of the development of the "distributed jugs" (d-Jugs) model.

6.2.1 The background of the case study

The jugs model (EMPA: jugsBeynon1988) is a single-modeller constructed EM model which resembles the real world situation in which two jugs can be filled with liquid, emptied, or poured from one to another (cf. figure 4.6 on page 112). The modeller can either interact with the jugs through the graphical user interface or by making redefinitions through the input panel of the tkeden. The jugs model is simple yet one of the classic models that EM researchers use to demonstrate the principles of EM. For this reason, it seems appropriate

[102] The first distributed jugs modelling session was collaboratively carried out by the thesis author and Antony Harfield. Part of that data generated in this case study has been used by Antony Harfield (2008) in relation to his doctoral thesis. In addition, the second distributed jugs modelling session has been partially presented in the WPCCS 2006 (Chan 2006).

[103] Although the modellers in the VEL project were building a distributed model cooperatively, there is no evidence that they exploited real-time collaboration though the DEM tool support (i.e. dtkeden).

to study how a variant of the jugs model can be built collaboratively.

The target variant was a "distributed jugs" model in which modellers situated at different workstations could get access to either of the jugs in the model. However, this is not a precise "specification" of the meaning of "distributed jugs" – this was interpreted slightly differently in the two collaborative modelling sessions. Nevertheless, the development of the distributed Jugs (d-Jugs) model is quite similar to the single modeller's jugs model, except the fact that modellers have to construct the jugs collaboratively and concurrently.

The d-Jugs modelling has been carried out several times with different pairs of modellers. This case study reports two instances of such modelling[104]. The first collaborative modelling session was carried out by the thesis author with Anthony Harfield, who was interested in computer-supported learning with EM. The second collaborative modelling session was the first part of a "collaborative modelling workshop" that I carried out with two students. The second part of the workshop – collaborative Sudoku – will be discussed in section 6.3. The primary aim of the workshop was to explore how modellers might practise EM collaboratively through building a EM model collaboratively in real-time in a single shared modelling space where the configuration is similar to *single display groupware* (Stewart et al., 1999). The secondary aim was to give the volunteering students a flavour of collaborative modelling with an EM approach[105].

6.2.2 Method of study and approach to data analysis

Although the d-Jugs modelling was carried out twice with two different pairs of modellers, the methods used in both modelling sessions were similar:

> *Minimal planning activities* – in both modelling sessions, modellers were asked to keep the planning of the modelling to the minimum. Moreover, the modelling time was short (2 hours approximately in each case) and neither of the modellers was

[104] Prior to these two instances of successful collaborative modelling, the d-Jugs modelling was not so successful due to technical issues in the tool support – and this technical issue was one of the motivations to develop better support for collaborative EM.

[105] This was used as a "strategy" to attract volunteers to participate in our study due to the fact that neither DEM nor collaborative modelling with EM was part of the "Introduction to Empirical Modelling" course.

briefed before the collaborative modelling sessions. In this way, the interactions between modellers throughout the modelling process are more likely to be unstructured and unplanned. This makes modellers more likely to *collaborate* (in the sense described in section 5.1.1).

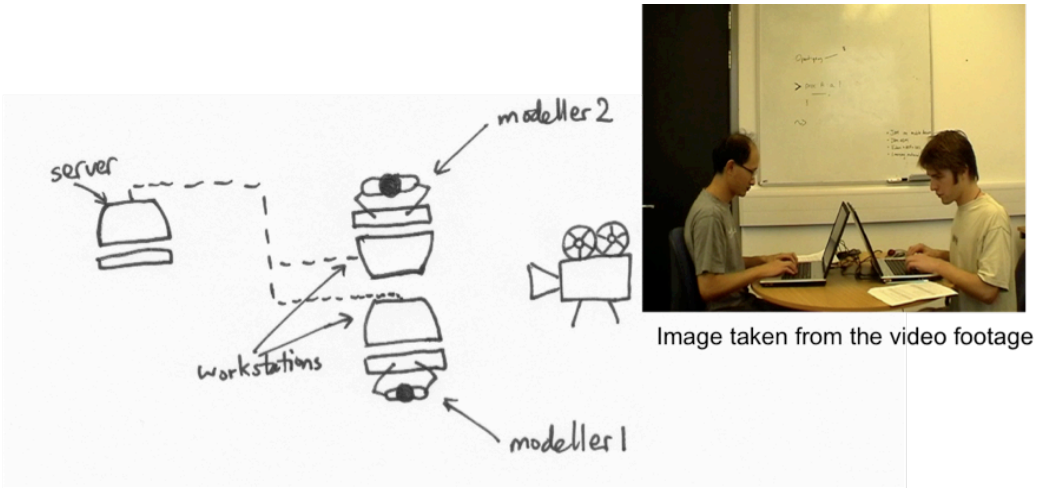

Figure 6.5 – The room configuration of the distributed jugs modelling case study

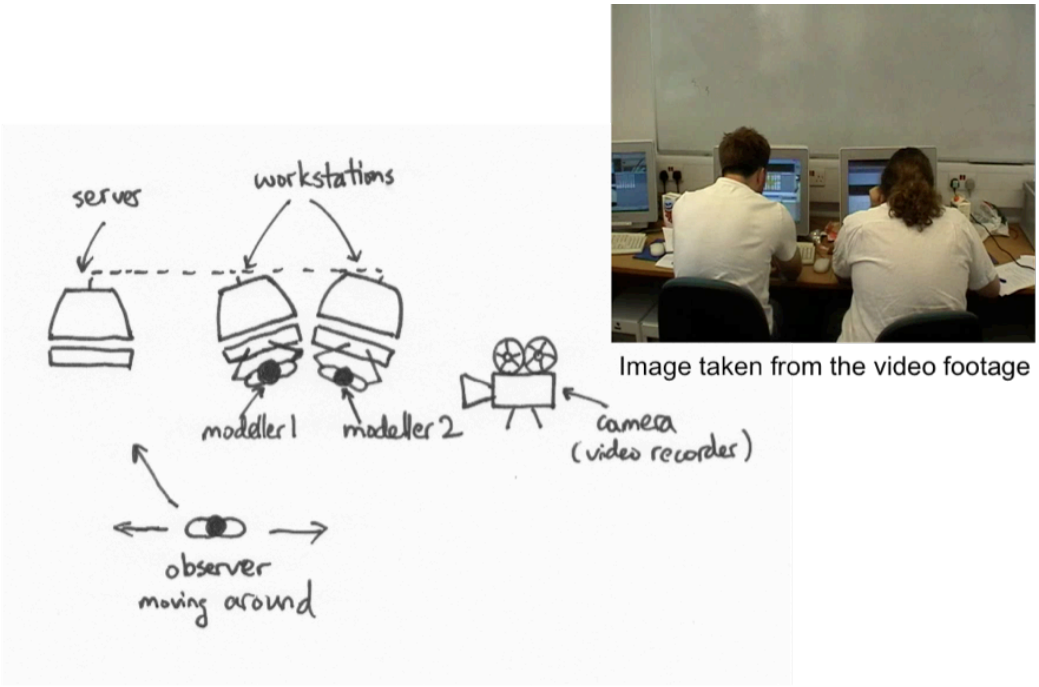

Figure 6.6 – The room configuration of the collaborative modelling workshop[106]

[106] Despite the fact that the camera was physically set up in the room, it was not used in the first part of the

Thinking aloud[107] – in both modelling sessions, modellers were encouraged to think aloud, i.e. to speak out what they are thinking when interpreting the meaning of what they see and their actions, why they are making such definitions or queries, etc. In this way, more information about the rationale or their state of mind can be obtained.

The data collection in both d-Jugs modelling sessions was informed by the participant observation approach (Spradley, 1980). Fieldnotes were taken on the spot during the modelling process. However, there was a significant difference in the degree of involvement between the first and the second d-Jugs modelling sessions. In the first d-Jugs modelling session, I was actively involved in the d-Jugs model construction while making observations to the modelling process. The degree of involvement was close to *complete participation* (Spradley, 1980). In contrast, in the second d-Jugs modelling session, I was merely involved in helping the modellers to troubleshoot the modelling environment when it was broken. Such a degree of involvement can be regarded as *passive participation* (Spradley, 1980).

Apart from the similarities in the study method, there are minor differences in the physical configurations and the tool support between the two versions of the d-Jugs modelling. In the first d-Jugs modelling session, the modellers sat face-to-face, with the workstations positioned in the middle between them (cf. figure 6.5). The modelling session were video recorded, screen captured (as video), and interactions within the modelling environment were logged.

In the second d-Jugs modelling session, the modeller pair were volunteering students selected from those who had attended the "Introduction to Empirical Modelling" course[108] in June 2006. The questionnaire that was used in the selection is attached in Appendix A. The main concern in the selection of students was availability and their proficiency in using tkeden, the primary tool support for individual EM. Neither of them had collaborative modelling experience with either dtkeden or dtkeden-cm prior to the d-Jugs modelling. The

collaborative modelling workshop (viz. the second distributed jugs modelling case study). It was only used in the second part of the collaborative modelling workshop (viz. the collaborative Sudoku case study, cf. 6.3).

[107] Cf. (Lewis and Rieman, 1994)

[108] "Introduction to Empirical Modelling (CS405)" is an optional module for the four-year MEng degree programme in Computer Science at University of Warwick.

physical configuration of the second d-Jugs modelling session was similar to that described in (Flor and Hutchins, 1992), where the modellers were sitting side-by-side (cf. figure 6.6). In contrast to the first modelling session, only interaction history was logged in the second modelling session. That is, there was no video recording and screen capturing. Two variants of dtkeden were used in the d-Jugs modelling sessions: in the first modelling session, the dtkeden-blackboard[109] was used; in the second modelling session, the dtkeden-cm[110] was used. The major difference between them is the mode of interaction. While the former supports the blackboard mode of interaction, the latter supplies an enhancement of dtkeden in "normal mode" with improved readability in the log files and different use scenarios (in contrast to the four closely associated with the DEM conceptual framework).

Despite the differences in the data collection techniques and physical configurations, both d-Jugs modelling sessions were analysed in a similar way. The motivation behind this strategy (i.e. analysing both modelling sessions in a similar way) was to make the results more comparable, so that cross-case analysis (cf. Yin, 2009) become possible. Both analyses can be roughly divided into four steps:

1. Reconstruct the modelling process by synthesizing a time-sequence log from multiple data sources.
2. Add annotations to the time-sequence log.
3. Identify events in the modelling process and seek explanation for the events.
4. Identify characteristics of the modelling process and seek explanation for the characteristics.

In the analysis of the first d-Jugs modelling, the time-sequence log was synthesized from

[109] This modified version of dtkeden was developed by the thesis author and Antony Harfield collaboratively for the purpose of the distributed jugs modelling. In dtkeden-blackboard, the virtual agency is partially removed so that it gives a single shared modelling space across all workstations and the server. This was later known as the "blackboard mode", as it resembles the blackboard metaphor (also cf. §5.1.3). Technically, the dtkeden-blackboard is equivalent to dtkeden in 'normal' mode with both automatic propagation and the virtual agency feature disabled, and some minor user interface changes.

[110] This enhancement gives the following features: (i) larger text in the input window (originally developed by Karl King); (ii) timestamps for the redefinitions logged in the interaction history file; and (iii) the distinction between definitions in the public space and in the private spaces was made explicit.

three data sources, namely, video footage, screen video capture, and the interaction history (in the form of definitive scripts). However, the synthesis of the data sources was far from straightforward. This was due to the fact that the interaction history has no timestamps. That is to say, I could gain no information about when a definition was input into the modelling environment from the interaction history alone. It was the screen video capture, after synchronising with the video footage, which provided the missing link between the video footage and the interaction history. In the second step, fieldnotes were edited and used to annotate the time-sequence log. Annotations were not by any means complete and they merely provided a starting point for further studying "what is going on?" in the modelling process. In the third step, the modelling process was broken down into events (or segments). The beginning timestamp, the ending timestamp, and the relative definitive scripts in the interaction history for the events were identified. Through multi-directional tracing between data sources, and with reference to the EM principles (cf. chapter 4), descriptions of the events and an explanation of the interactions between the modellers were derived in the context of collaborative modelling (cf. chapter 5). Attention was also given to the social context in which the collaborative modelling took place, e.g. the use of and the switching between the private and the public modelling spaces. To avoid bias in the analysis of the events, consideration were given to whether the modellers were actually practising an EM approach. In the fourth step, I looked at the modelling process at the holistic level. Attention was given to the role of EM and its tool support in the collaborative modelling process.

Similar analysis techniques were used in the analysis of the second d-Jugs modelling. However, there were two significant differences between the analysis of the first and the second d-Jugs modelling. Firstly, the time-sequence log of the second d-Jugs modelling was synthesized from two data sources, namely, video footage and the interaction history. This was because the screen video capture was not possible due to a technical limitation. Secondly, the data set for analysis was much larger than in the first d-Jugs modelling session. This was due to the fact that the modelling process in the second d-Jugs modelling session was much longer.

6.2.3 Limitations

In the distributed jugs case study, both d-Jugs modelling sessions were informed by participant observation (Spradley, 1980). In the first d-jugs modelling session, complete participant observation was adopted as one of the techniques for data collection. It enabled me, as a researcher, to obtain deep insight into the process of collaborative modelling with an EM approach. Such first-person experience would be otherwise difficult, if not impossible, to obtain without being a participant inside the activity.

The major criticism for participant observation is that the study may be shaped by the researcher's bias, whether intentionally or unintentionally. It is hard to avoid such criticism even if the researcher has consciously tried to avoid bias. To address this criticism, a reduced degree of involvement (i.e. passive participant observation) was adopted in the second d-Jugs modelling session. That is, help was only offered to the modellers when they struggled with the tools and were unable to overcome an issue in a timely manner. Moreover, other means of data collection were also used in both d-Jugs modelling sessions. Having multiple sources of data is an advantage of case study as a research method (Yin, 2009). To some extent, this has enabled data triangulation (Yin, 2009), so that the findings of the case study do not rely solely on the fieldnotes that were generated from participant observation.

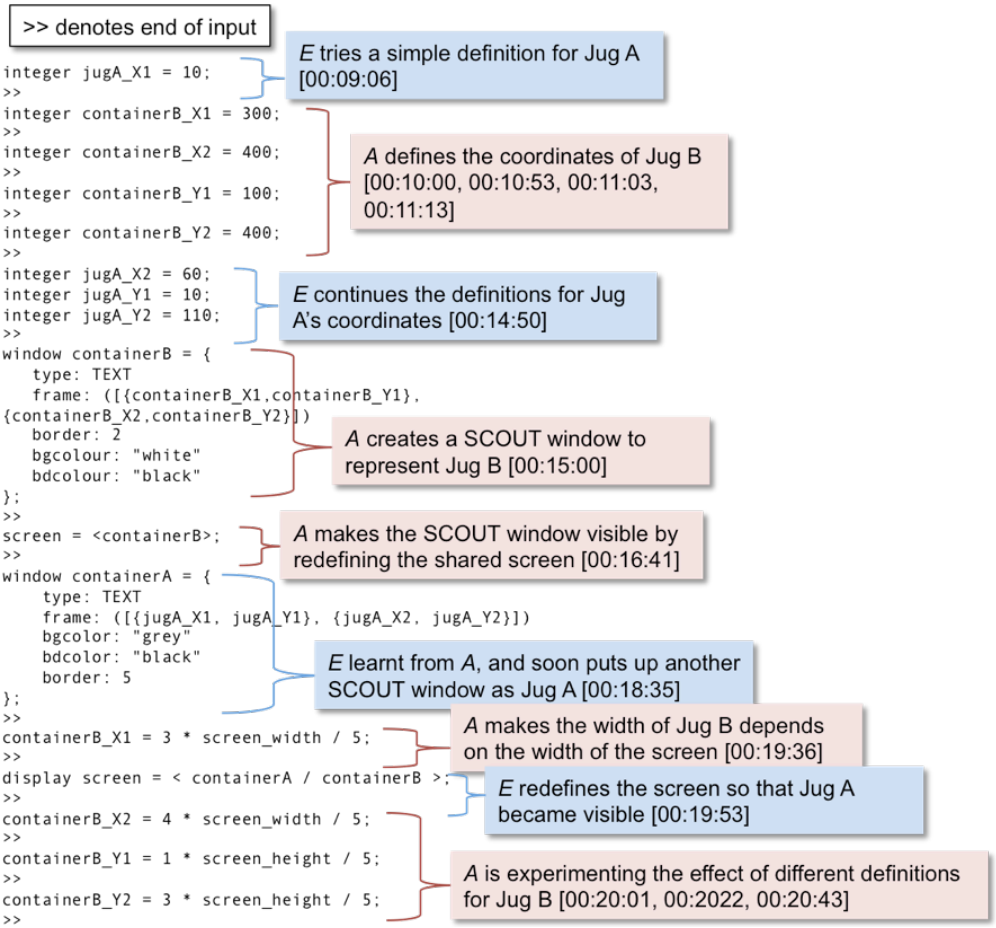

Figure 6.7 – A glimpse of the interaction between two modellers in a distributed jugs modelling[111]

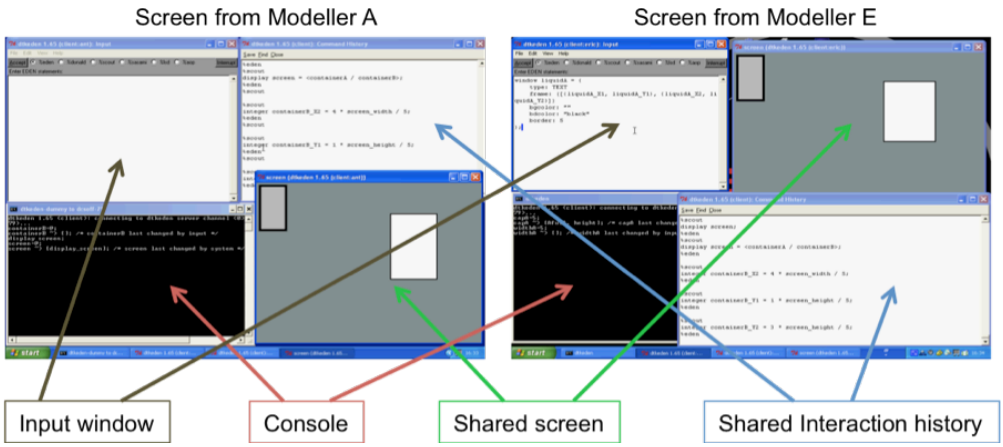

Figure 6.8 – Collaborative modelling of a pair of jugs at the early stage

[111] This figure is modified from figure 5.24 in (Harfield 2008, p.170), with an improved description of the actions by the modellers and timestamps added (for cross referencing with figure 6.9)

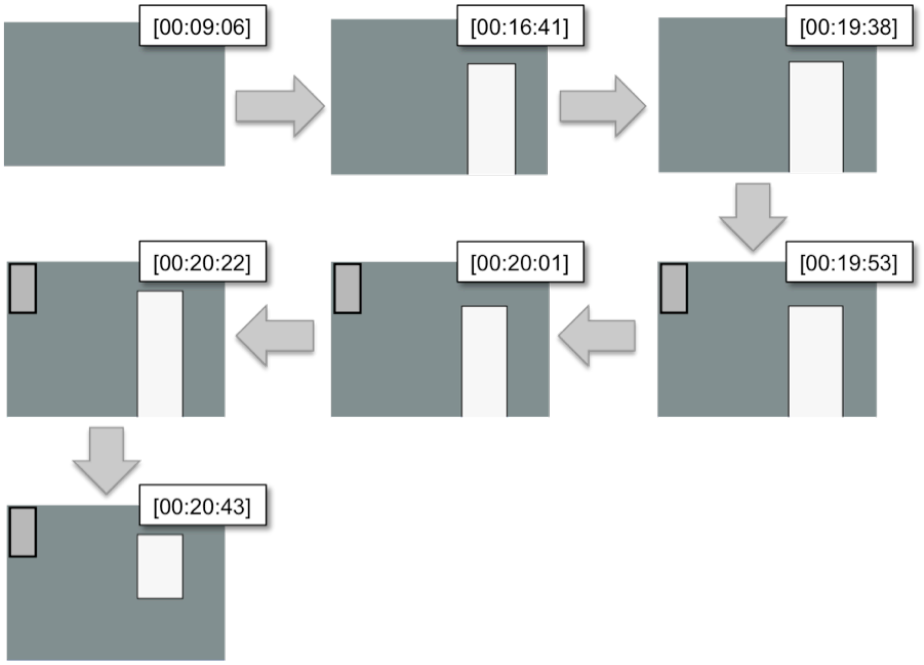

Figure 6.9 – A glimpse of the development of the distributed jugs in a collaborative modelling environment

6.2.4 The collaborative modelling process

In the first d-Jugs modelling session, modellers did not begin the collaborative modelling from scratch. After some discussion, the text-based jugs model[112] was adopted as the foundation for the d-Jugs model. Indeed, the aim of the modelling session was to study the collaborative modelling process, not to construct the product (i.e. a working, polished jugs model). Despite the fact that the modellers in this modelling session were very experienced in using tool support for EM, they were struggling with different sorts of problems (e.g. invalid syntax, being unaware of the current notation) during the first 9 minutes of the collaborative modelling process. Instead of making complex definitions, which these modellers normally do when they practise EM on their own, they merely made simple one-line definitions such as *"capA = 5;"* and *"Aempty is contentA==0;"* initially. Later, modeller *E* made a breakthrough by making a simple definition for the coordinates of Jug A (cf. figure 6.7). Soon, taking up modeller *E*'s suggestion, modeller *A* made a few definitions for the

[112] A variant of the original jugs model (EMPA: jugsBeynon1988) that was used in a laboratory of the "CS405 Introduction to Empirical Modelling" course in the academic year 2006/07.

coordinates of Jug B. Then, modeller *E* completed the definition for the coordinates of his jug. After the initial success, modeller *A* went back to what he did initially – defining a SCOUT window to represent a jug – and made that visible on the shared screen. At this moment, modeller *E* got excited – since he saw a jug appearing on the shared screen. He created another SCOUT window to represent his jug, by referring to the definition of Jug B[113] created by modeller *A* a moment ago (cf. figure 6.7 and the second screen in figure 6.9). On the other side, modeller *A* was not quite satisfied with the appearance of his jug, and experimented with the effect of different definitions. Figure 6.9 highlights the crucial moments – in relation to visual experience on the shared screen – in this short collaborative modelling session.

In the second d-Jugs modelling session, modellers were asked to extend the jugs model (a model used in a laboratory of the "Introduction to Empirical Modelling" course in 2006)[114] into three jugs. Their collaborative modelling process can be summarised as follows[115]:

1. In contrast to the first d-jugs modelling session, the modellers were testing the collaborative modelling environment with simple definitions during the first 15 minutes (c.f. Appendix D1). This is probably due to the fact that the dtkeden-cm environment was new to them, and the modellers wished to find out the difference between "Accept Locally" and "Send to Public" (cf. Appendix B).

2. After the initial testing, the modellers soon switched to composing their scripts in an external editor, copying and pasting across from the editor to the modelling environment. For instance, at one point, modeller *R* repeatedly copied and pasted a similar script a few times from his external editor to the modelling environment (c.f. Appendix D2)[116].This is another style of modelling,

[113] The SCOUT definition of Jug B was displayed in the shared interaction history window located in the lower right-hand corner of his screen (cf. figure 6.7)

[114] A variant of the original jugs model (EMPA: jugsBeynon1988) that was used in a laboratory of the "CS405 Introduction to Empirical Modelling" course in the academic year 2006/07.

[115] The collaborative modelling process is reconstructed mainly based on the interaction history logged on the workstations.

[116] The similarities among (re)definitions and the short time intervals between (re)definitions suggests that modeller

unlike that used by the modellers in the first d-Jugs modelling session. In contrast, their "interaction" within the modelling environment became "repetitive" because of their copy-and-paste.

3. Then the modellers began to work in their private workspaces and occasionally put their mature pieces into the public space (cf. Appendix D2 and D3). For instance, modeller *S* experimented for about 10 minutes in his private space with different sizes and colours for the SCOUT window *fillA* (which represents a button in this model) before he put that button, together with another two buttons (*pourAtoBbutton* and *pourAtoCbutton* – developed by cloning and modifying the *fillA* button – into the public space (cf. 10:42:54 to 10:54:52 Appendix D3).

4. For the first 25 minutes after initial testing, modeller *R* had been working privately in an editor (external to the modelling environment). Then, he turned to test his piece in his private modelling environment. When that piece (including some definitions for Jug C, all the button agents, pouring agents and the animation agent) matured, he put it into the public space [117].

5. Modeller *R* tested his piece in the private space again, making sure the buttons were working[118]. He then put all the dependencies for guarding the button in the public space simultaneously. Again, modeller *R* put his piece first into the private space (to make sure that will not cause any harm in the public space),

---

R was not typing the (re)definitions but modifying them in an external editor and copying-and-pasting them to the modelling environment.

[117] No (re)definitions of Jug C were sent to the public space between 10:30:39 (the last redefinition from modeller S that was captured on the dtkeden-cm server) and 10:54:38 (the first definition that modeller S sent to the public modelling space after the initial testing). The latter definition indicates the end of the initial testing (cf. Appendix D1). Meanwhile, the definition that modeller R sent to his private modelling space at 10:55:43 indicates that modeller R was beginning to test his contribution to the (re)definitions of Jug C in his private modelling space (cf. Appendix D2).

[118] Modeller R did not define the SCOUT windows for the buttons – it was modeller S who defined the buttons (SCOUT windows) *fillA*, *pourAtoBbutton*, and *pourAtoCbutton*. (cf. Appendix D3). Rather, modeller R defined the trigger procedures behind the buttons (cf. 10:58:02 to 10:59:49 in Appendix D2). The interaction history between 10:58:02 and 10:58:03 (cf. Appendix D2) indicates mouse button clicks on the SCOUT windows: *fillA*, *pourAtoBbutton*, and *pourAtoCbutton*. Though it was not 'formal' software testing, these mouse clicks suggests that modeller R was experimenting or 'playing around' with the buttons after he sent the Jug C definitions to the public modelling space.

then into the public space (cf. 10:59:44 and 10:59:49 in Appendix D2).

6. While modeller *R* was busy in putting Jug C into the shared modelling space, modeller *S* was testing the other two jugs – Jug A and Jug B – in his private modelling space. The interaction history between 10:55:43 and 10:58:03 shows that modeller *R* was concentrating on making the basic definitions for Jug C and making the pouring buttons work (cf. (2) above and Appendix D2). Meanwhile, the interaction history between 10:42:54 and 11:06:15 shows that modeller *S* was concentrating on making definitions for the SCOUT windows that represents various buttons for the filling function for Jug A and Jug B, and the pouring actions between them (cf. Appendix D3 and D4).

7. Soon, it became apparent that the construction was pretty much dominated by modeller S, and modeller *R* shifted into a more passive role. This is to say, the collaborative modelling of the distributed jugs was largely driven by modeller S. This is suggested by the observation that modeller *R* did not make any contribution to the public modelling space between 10:59:49 and 11:30:25 (cf. Appendix 5).

8. At around 11:40, the dtkeden-cm client on modeller *R* was malfunctioning. The pair restarted the dtkeden-cm server and clients, and reloaded all the scripts that they had backed up in their private editors.

9. Mature versions of Jug A, Jug B, Jug C, and all the GUI agents were developed after 85 minutes of collaborative modelling (cf. Appendix D6).

The data excerpt in appendix D6 shows the major contributions from the modellers which have shaped the final version of their d-Jugs model in the modelling session. Despite the fact that their d-Jugs model could perform some basic operations (e.g. pouring from one jug to another), there are some 'bugs' in their model. In order to make their d-Jugs model function as expected, the definitions that are listed in appendix D7 would be required. Figure 6.10 shows a screen capture of the resulting d-Jugs model, i.e. the final version of the d-Jugs model constructed by modellers R and S with the definitions in appendix D7 added.

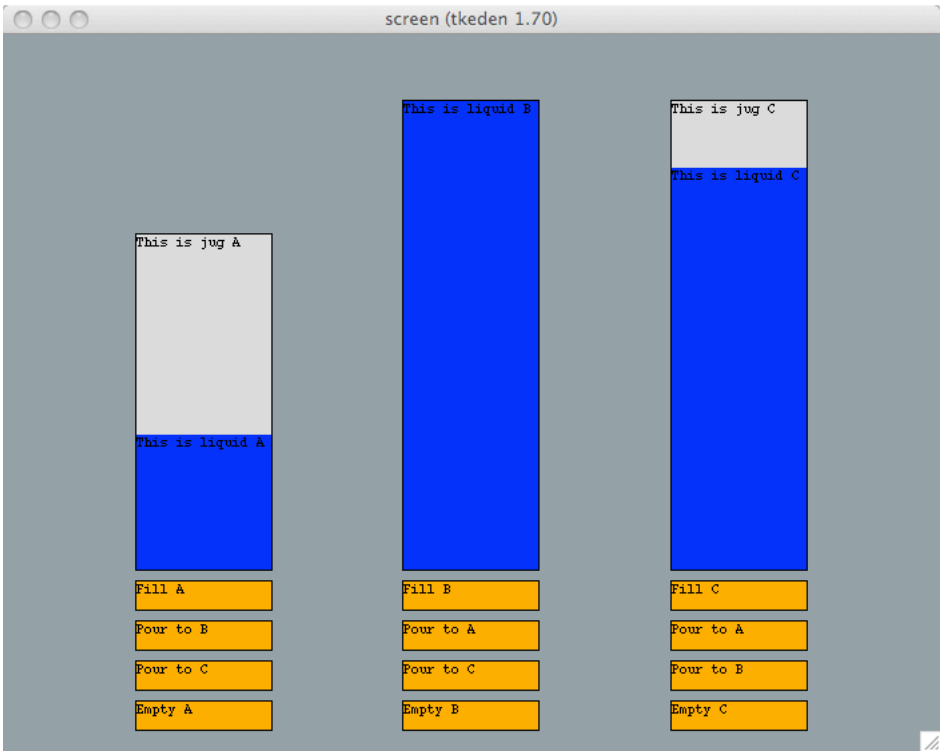

Figure 6.10 – Screen capture of the distributed Jugs model constructed in the second distributed jugs modelling session

6.2.5 Discussion

Although both d-Jugs modelling sessions are trivial and short, they do illustrate some issues and potential benefits of practising EM in collaborative modelling.

Coordination in the shared space

In the first d-Jugs modelling session, both modellers are experienced in tool support for EM. Both modellers are familiar with the text-based jugs model (EMPA: jugsBeynon1988). Consequently, there was no planning at all; responsibility for the observables, dependencies, and agents were negotiated *in situ*. For instance, at the beginning of the modelling process, modeller *E* said that he would like to construct the graphical representation of Jug A on the screen. Modeller *A* responded to modeller *E* that he would create the other jug (Jug B). Modeller *A* and modeller *E* had a short conversation about where to put their jugs before they introduced their first definitions relating to their jugs. In the second d-Jugs modelling session, the modellers simply avoided conflict by working in their private editors and only using the shared modelling space when their pieces were

mature.

Despite working in a shared modelling environment, modellers in both d-Jugs modelling sessions have actively avoided the potential conflicts in changing others' work, through different strategies. In the first case, modellers were using preceding verbal communication and queries to the identifier in question[119]. The dtkeden-blackboard (and dtkeden-cm) modelling environment responds to queries of identifiers promptly[120]. For instance, Modeller A queried the *screen* observable[121] before he made a definition in relation to it.

To some extent, this mode of working may facilitate collaborative modelling effectively but unobtrusively. In the second case, modellers exploited the private and public spaces. Careful analysis suggests that these are two different kinds of collaborative modelling; while both can be regarded as exemplifying the MMMA scenario, the former is *collaboration* and the latter is *cooperation* (cf. the discussion in section 5.1.1). On the one hand, this seems analoguous to the natural avoidance of entering another's "personal territory" in spatial interaction (cf. Scott et al., 2004; Tse et al., 2004), e.g. using whiteboards or tabletop groupware. On the other hand, this suggests that EM coped well with two degrees of engagement in collaborative modelling.

Experimentation

In the first d-Jugs modelling session, modeller *A* was not satisfied with the appearance of his jug. He made use of the experimental characteristics of the EM tool support to *try-and-feel* the effect of different definitions. Similar *try-and-feel* actions are also found in the second d-Jugs modelling session, where both modeller *S* and *R* experimented with their pieces before making them available to the other modeller via the shared modelling space. This suggests that the experimental characteristics of EM may play an active role in real-time collaborative modelling, where modellers want to verify their perceptions (i.e. their observation of the

[119] That is, to check whether it has been used and, if so, consult its current definition before making any changes. An identifier in tkeden and all its variants (including dtkeden-blackboard and dtkeden-cm that were used in the modelling sessions in the case study) can be an observable, a dependency, or an agent.

[120] This feature is in fact inherited from tkeden, the primary tool support for EM.

[121] An observable which is often used to denote the default SCOUT screen in most EM tool support.

referent in the sense of EM) either through public experimentation – in the first case – or through private experimentation – in the second case. Apart from potential conflicts, when well coordinated, the only difference between public and private experimentation is its visibility.

Role-shifting

In chapter 3, I argued that multiple roles and shifting roles of participants feature in the systems development process. If we view the collaborative modelling process as a miniature form of a systems development process, and attribute roles to modellers in relation to their task in the situation and their current focus of attention, it is not difficult to imagine a modeller may have multiple roles and shifting roles throughout the collaborative modelling process. In other words, a modeller can be thought as a developer, a learner, a user, a teacher, a demonstrator, etc. The d-Jugs modelling sessions seem in line with this phenomenon. In the first d-Jugs modelling session, modeller *E* observed, learnt and cloned SCOUT definitions of a jug from modeller *A* (cf. figure 6.7 on page 187, 00:18:35). In that moment, modeller *E* was in a learner role rather than merely a developer. In the same spirit, modeller R, in the second d-Jugs modelling session, has turned into a learner and modeller *S* became a teacher (or demonstrator) in the later stages of collaborative modelling process. In addition, both modeller *S* and modeller *R* verified their definitions in their private modelling spaces before they made them public. Such behaviour can be thought as *using* a component of the model in their private space. Taking all the above into consideration, it is plausible to claim that EM does not obstructed the role-shifting phenomenon and, to some extent, its open, flexible, experimental, and instant feedback characteristics facilitate seamless shifts.

Dialogical and diverse interaction

As I argued in section 4.3, the interaction between the modellers and the interaction between an individual modeller and the agents within the model (whether in the private or public modelling space) can be thought of as dialogical. As shown in figure 6.7 (on page 187), the interaction between modellers *A* and *E* more or less resembles a dialogue (or a conversation). This resemblance stems from the pattern of interaction, whereby A and E

take it in turns to contribute to the definitions of Jug A and Jug B, and the common theme of their interactions.

The effect of this dialogue is to negotiate the meaning of "distributed jugs" (cf. §6.2.1) through redefining the SCOUT screen. For instance, modeller *A* could have defined:

```
screen = <containerA / containerB / liquidA / liquidB>
```

but instead, he actually defined:

```
screen = <containerB>
```

overlooking the fact that there should be several SCOUT windows on the screen. It was modeller *E* who spotted the issue a moment later when he could not see two jugs on the screen. Even then, *E* did not consider the need for more SCOUT windows (viz. *liquidA* and *liquidB*), which he would have to add subsequently. Similar dialogical interaction can be found in the second d-Jugs modelling session, where modeller *S* and modeller *R* exchanged definitions through the public modelling space and interacted with their local model in their private modelling space through redefining observables. The interaction among all parties (i.e. the modeller, and the agents in the model) suggests that collaborative modelling, when practising an EM approach, is dialogical.

In both d-Jugs modelling sessions, modellers interacted through different channels, which included the evolving EM model, gestures around the screen, and verbal communication. They also switched between different means of communication seamlessly to: i) exchange their perspectives; ii) negotiate when conflicting perspectives arise; iii) coordinate their work; iv) inform about and diagnose each other's progress. The virtue of the blackboard mode is that it allows diverse modes of interaction, which as mentioned earlier also enables the modeller to shift their roles seamlessly.

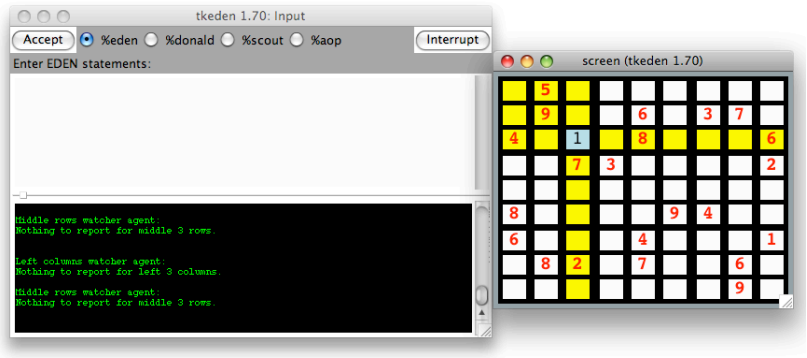

Figure 6.11 – A screen capture of King's Sudoku model (EMPA: sudokuKing2006) with the tkeden modelling environment

6.3 Case study III: collaborative Sudoku

Like the second d-Jugs modelling session described in the previous section, the collaborative Sudoku modelling was part of the collaborative modelling workshop. However, unlike the case study of the d-Jugs modelling, the focus for the model building was more complicated than in the previous one, and related to computer support for human solving of Sudoku puzzles.

6.3.1 The background of the case study

Sudoku is a popular puzzle game in newspapers and magazines[122]. The first EM model of the Sudoku puzzle was constructed by Karl King (EMPA: sudokuKing2006) in relation to his Masters thesis (King, 2007). For King, the modelling of the Sudoku was:

> *"an exercise in embodying personal understanding within a computational artefact in conjunction with skill development" and "it was to explore ways in which the puzzle solving exercise could be usefully supported by the computer, and how the skills of the human solver could be embodied in a computer-based artefact" (ibid, p.48).*

As depicted in figure 6.11, King developed a number of visual aids to assist the human solver of the original Sudoku puzzle game. King's Sudoku model was enhanced by

[122] http://www.guardian.co.uk/media/2005/may/15/pressandpublishing.usnews

Efstathiou (2006) for studying his definitive notation for representing combinatorial graphs (C-Graph) (also cf. EMPA: sudokuEfstathiou2006) and later enhanced with colours by Harfield for tackling Sudoku with a novel technique (EMPA: sudokucolourHarfield2007). The collaborative Sudoku modelling was based on King's Sudoku model, and it was made available to the modellers at the beginning of the modelling session.

6.3.2 Method of study and approach to data analysis

As a part of the collaborative modelling workshop, the collaborative Sudoku modelling has the same physical configurations of the equipment, where modellers were adjacently seated and working on a shared model through separated but networked workstations (cf. figure 6.6). The modeller pair was the same pair who participated in the first part of the collaborative modelling workshop. The modelling session was video recorded and interaction within the modelling environment was logged. As in the first part of the collaborative modelling workshop, dtkeden-cm was used in the modelling session (cf. §6.2.2). Due to instability of the tool support, the observer had an active role in supporting the modellers when the modelling environment broke down[123], though this was a rare occurrence.

During the collaborative modelling process, the modellers were asked to develop a collaborative strategy to tackle any given Sudoku puzzle collaboratively. Our intention was to encourage the modellers to make use of observables, dependencies, agents, and agencies. For instance, the modellers were told that they could represent the way in which the value in one square affects another using dependencies, and that through negotiating and validating the dependencies the solution may emerge. They were also given prototypes for the agents that would exploit such dependencies to implement rules of the form "if digit *i* is placed in cell *x* then digit *i* must also by placed in cell *y*". The nine agents for the digits (viz. *i* equals 1 to 9) were called *'checkrules1'* to *'checkrules9'* respectively.

In addition, they were encouraged to make use of the benefit of having a separate private

[123] To avoid unnecessary interference to the modeller, we only provide support when there was a breakdown in using the tool support and when they could not remedy the situation during the modelling session.

space for their local experimentation when necessary. Apart from those guidelines, the modellers were free to make any changes within the modelling environment, free to work in any style, and free to switch to any modes of interaction when necessary.

The analysis strategy for the collaborative Sudoku case study was similar to the four-step analysis strategy that was adopted in the distributed jugs case study (cf. §6.2.2). The topical research question for the analysis was "how do the modellers practise an EM approach to collaborative modelling collocated at real-time?" Therefore, the main focus was on building an explanation for such a modelling situation. In comparison with the distributed jugs modelling case study, collaborative Sudoku generated significantly more data for analysis. Despite the fact that timestamps were logged in the interaction history (an improvement in data collection over the distributed jugs case study), the synthesis of the data sources in the analysis of the collaborative Sudoku modelling was not as trivial as expected. This was due to the fact that the interaction history were not originally designed solely for research purposes. Consequently, they were scattered in different locations and overlapped in some respects. As in the analysis used in the distributed jugs modelling case study (cf. §6.2.2), an annotated time-sequence log was generated after the second step of the analysis. Then, events in and characteristics of the modelling process were identified based on the annotated time-sequence log, the video footage, and the interaction history.

Apart from the similarities, the analysis for the collaborative Sudoku case study was different from the four-step analysis strategy that was adopted in the distributed jugs case study in the following ways:

i. *Dialogue transcription from the video footage* – Due to the complex interaction that was observed in the events, all the dialogues in the video footage were also transcribed literally. These transcribed dialogues were used as supporting evidence for the claims in the discussion section (see §6.3.4 for details).

ii. *Informal interview at the end of the modelling session* – During the collaborative Sudoku modelling, I observed that the modellers were struggling to collaborate efficiently. At that point, it was unclear to me what caused the issue. For this reason, I asked the modellers to give their feedback at the end of the collaborative Sudoku

modelling.

6.3.3 Limitations

The collaborative Sudoku modelling session was the second part of the Collaborative Modelling workshop, which aimed to reveal how EM modellers approach and collaborate in a more complex problem situation. As in the second d-Jugs modelling session, the collaborative Sudoku case study adopted passive participant observation as one of the techniques for data collection. Likewise, other means of data collection (viz. video capture, and interaction history) were used to avoid some of the pitfalls of participant observation. The main reason that participant observation was not rejected is that it gives an opportunity for the researcher to rectify minor problems, such as tool stability in our case (Yin, 2009). It would otherwise be difficult for the researcher to rectify the problems as an external observer.

6.3.4 Discussion

The original intention of the collaborative Sudoku modelling was to reveal how modellers might carry out EM collaboratively with two modelling spaces – their private modelling space and a shared modelling space – at real-time. That is, to allow the modellers to improve the Sudoku model collaboratively through reflecting on their own experience[124] in relation to the game, and through interaction with the model and the other modeller. The idea was to reveal the interplay between the developing artifact and the modellers' understanding throughout the modelling process.

The pair of modellers in the collaborative Sudoku modelling was the same pair who participated in the second d-Jugs modelling session (cf. §6.2). As in the d-Jugs modelling session, the modellers in the collaborative Sudoku modelling exhibited similar role-shifting behaviour, dialogical interaction, and diverse interaction behaviour. However, the result of the collaborative Sudoku modelling was quite different in relation to the artifact – there was no "working product" out of the collaboration process. Despite the fact that Sudoku was so

[124] That is, the puzzle is printed on a paper and solved using a pencil and a rubber.

popular, the modellers had not heard of the Sudoku. In other words, they had no previous experience to draw on. It became a "collaborative comprehension exercise", and incidentally made the modelling process similar to the early stage of systems development where participants are spending much of their time in learning the problem domain[125]. It turned out to be an interesting scenario because the modellers were not only co-constructing an understanding of that problem situation, but because that understanding and the goal of the activity also co-evolved through the modelling process.

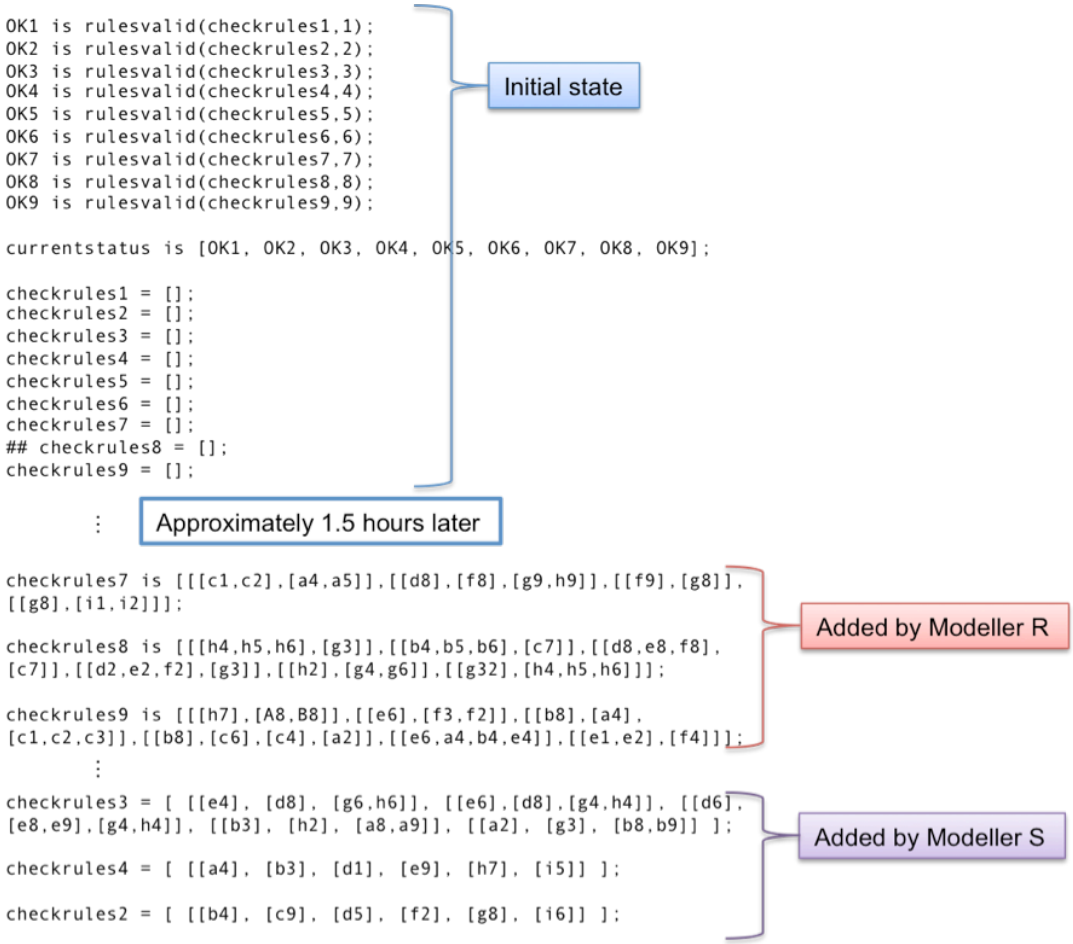

Figure 6.12 – Co-construction of the check rules in the collaborative Sudoku model

Co-construction and co-evolution

[125] Apart from the scale, our situation here is similar to the situation in Danielsson's (2004) mobile learning systems development project, where the students (i.e. future users) of the system had no knowledge of what could be done with the technology and therefore the students became learners in the early stage of the systems development process.

The first one and a half hours of the collaborative Sudoku modelling can be thought of as a co-construction through EM of a shared understanding (e.g. what Sudoku is about, and what is the collaboration strategy for tackling the puzzle) between the modellers. In this period, the modellers gradually understand what the "check rules" can do, as a mechanism to tackle the puzzle, and what is its limitation. They also managed to define the "check rules" for some numbers and at least agreed that the check rules for digit 9 "seems fine". As shown in figure 6.12, the pair co-constructed some "check rules" after approximately 1.5 hours of collaboration. However, the modellers did not leap from the initial set of check rules into the final set at once. Taking the set of check rules for the number eight (i.e. *'checkrules8'* agent) as an example, modeller *R* refined *'checkrules8'* in all 13 times throughout the modelling process (cf. figure 6.13). Similar updates were made to all the other *'checkrule'* agents in the model. As we discussed in chapter 4, all agents co-evolve with the modeller's understanding of the situation in EM. Indeed, any redefinitions of '*checkrules8*' would also cause immediate redefinition of other observables in the model (e.g. *'currentstatus'*) and that in turn affected modeller S's action in redefining other *'checkrules'*. This interplay between modellers and redefinition exemplified one small aspect in the negotiation of the meaning of possible solution (of the puzzle).

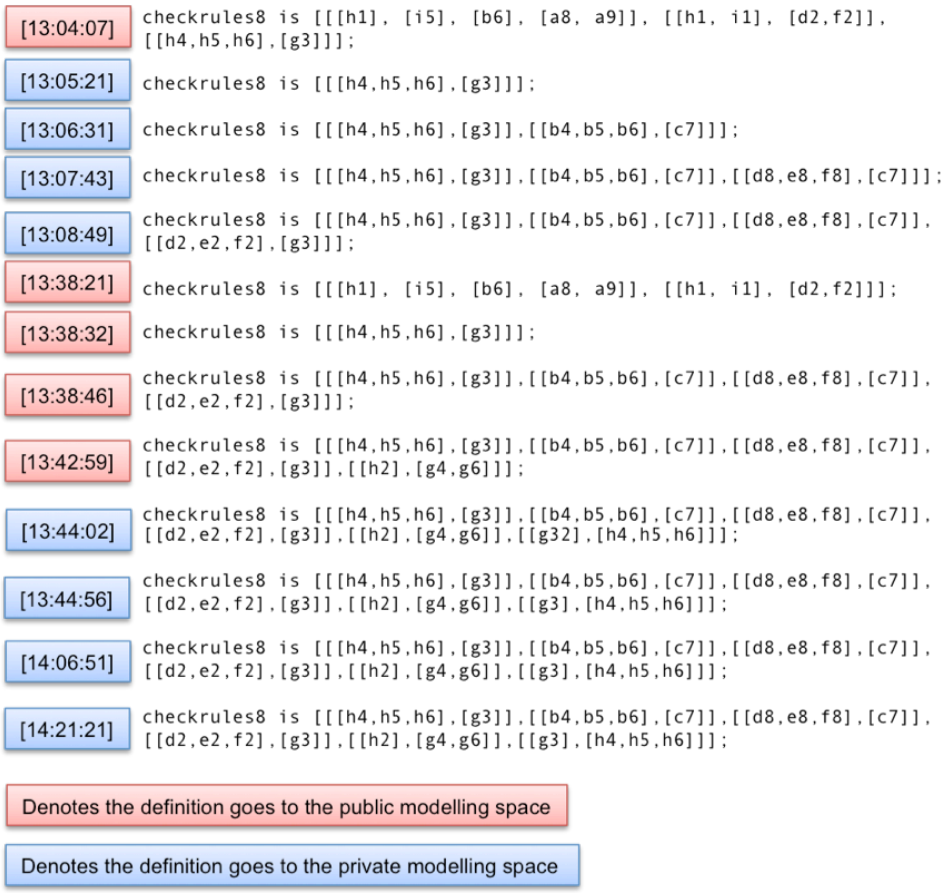

Figure 6.13 – Evolution of a set of check rules in the collaborative Sudoku model

```
proc wcs: currentsquare {
     write(currentsquare);
     write(" / ");
     write(currentstatus);
     write(" / ");
     writeln(current_region // current_row // current_column );
}
```

Figure 6.14 – A little helper agent 'wcs' used in the collaborative Sudoku modelling

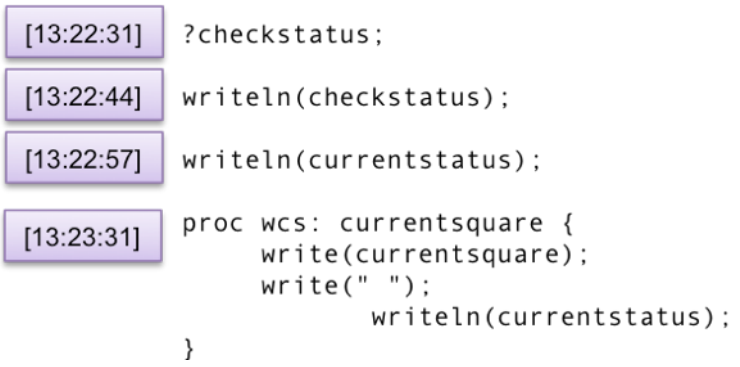

Figure 6.15 – Modeller S queried the 'checkstatus' observable before defining the initial

version of the little helper agent 'wcs'

Despite the short time span of the collaborative Sudoku modelling, there is some evidence that the goal of the modelling process had evolved from "developing a collaboration strategy for tackling the puzzle" to "tackle the puzzle first!" The goal shifting was first observed when modeller *S* focused on tackling the puzzle rather than further development of the check rules[126]. He seemed to be enjoying the game play but the goal shifting (as well as role-shifting – from a developer of the model to a user of the model) became apparent when modeller *R* noticed that his colleague has changed his role, while he was struggling with making progress for a set of check rules for the number 9. Then, modeller *R* joined in the solving process, and at that point (1.5 hours after the collaborative modelling began), the shared goal became "tackle the puzzle first!"

Evolving collaboration support

There is evidence that the modellers not only co-constructed the check rules for the Sudoku model, but also co-evolved the collaboration support for the quest. For instance, modeller *S* made an agent (in the form of a triggered procedure) to show the coordinate of the square and the region where the mouse is located. A moment later, modeller *S* shared that agent with modeller *R* upon his request. Then, modeller *S* further developed the little agent '*wcs*' for the next 7 minutes (with input from modeller R) before he finalised it as shown in figure 6.14. This was all initiated by the conversation:

[126] It was the moment right after modeller R's workstation crashed.

Modeller S: have you got any cheap and nasty way to show whether the status has changed?

Modeller R: [delay a few seconds – because he was busy doing something] Err.. Whether what status has changed?

Modeller S: The … is it, is it checkstatus went to the main yea … [making a query "?checkstatus" to dtkeden-cm (cf. figure 6.15)] enable the '1's there … what sort of list of the list of ones?

Modeller R: Err… yea, write line on check status.

Modeller S: [entering "writeln(checkstatus);" to dtkeden-cm (cf. figure 6.15)] Humm … hu-er hu-er … [got "@" (undefined) from dtkeden-cm leaning over to richard's screen] just check, just check …

Modeller R: ok, that? … current status.

Modeller S: current status.

Modeller R: Right do.

Modeller R: Umm…

Modeller S: Right, ok.

Modeller R: Can you pop in there … ?

Modeller S: What I'll do is I'll pop that in a bit with mouse over.

Indeed, the little helper agent (cf. figure 6.14) played an important role in the collaboration – it provided a shared reference for the modeller to communicate[127]. On another occasion, modeller *S* redefined all the agents behind the squares, so that any changes to the values of the squares would become a public redefinition, whether it is made in the public or private modelling space. This was in response to the need for better collaboration support. More importantly, this provides evidence that the instant feedback characteristic of EM can be easily converted into real-time collaboration support. In addition, such ad-hoc evolution might be difficult in the context of traditional software development, but requires relatively little effort when practising EM.

Collaboration and conflation of the contexts of development and use

As mentioned in chapter 4, the practice of EM potentially conflates the contexts of development and use. In the collaborative Sudoku modelling, it is hard to distinguish the

[127] In principle, it is possible that such a helper agent might evolve into an awareness support agent, which provides information about the current focus of attention of the other modeller.

context of use from the context of development, as both modellers are actively 'using' the Sudoku model like a normal player while thinking about how the check rules or other agents can be developed as described earlier.

Issues with the tool support

The collaborative Sudoku modelling also highlights several issues in the current tool support for collaborative modelling. Firstly, it seems that the tool itself is not stable enough for an extended collaborative modelling exercise. The server crashed twice and the clients were required to restart several times due to malfunctioning during the collaborative Sudoku modelling. Instead of the input panel provided by the tool support, both modellers used external text editors as their temporary workspaces. They believed that their approach would give extra reliability (i.e. to keep their script safe) and security (i.e. to reload the script in a timely way when the tool crashed). Secondly, the transition between private and public workspace is not without issues. For instance, modellers often need to interrogate the modelling environment (i.e. to make a series of queries concerning the same observable) to determine whether the definition of the observable is part of the model in the public modelling space or part of the model in the private modelling space. Besides, they often asked the other modeller to share his work in the public space. To some extent, this hinted that the blackboard mode might be better for synchronised collaboration due to its single shared space[128]. Thirdly, it is impossible to switch to other modes of interaction from the client side. The tools that I used in the collaborative modelling sessions (i.e. the d-Jugs modelling and the collaborative Sudoku modelling) implemented one mode of interaction at a time. For instance, dtkeden-blackboard is for blackboard mode, dtkeden-cm is a combination of the blackboard mode and the private mode. Nevertheless, the current architecture of the dtkeden requires the switching to be done on the server. At the moment, dynamic and client-determined modes of interaction are not possible without re-implementation of the tool.

[128] However, as discussed in chapter 2, people's information sharing behaviour can be quite subtle. Hence, the single shared space may not work for all groups.

The difficulty in coordination

At the end of the collaborative modelling workshop, I had an informal discussion with the modellers about their feeling of the difficult situation they had faced in the collaborative Sudoku modelling. They believed that tackling a Sudoku puzzle, as well as developing a strategy for it, is a difficult problem situation because there is no easy way to partition the problem. This is reflected in the following conversation:

> Modeller S: It's difficult with this one to know what you can … add to the model which is going to significantly to help you which is kind of worth the effort.
>
> Modeller R: The problem with the check rules was that you could have … it was, it wasn't care whether number is correct. There's usually a problem, is it? Like you could place them, you could place how nine each of the numbers so they match themselves fairly easily. It's one of the check rules, kind of the crossover, that we're having … problem.
>
> Modeller S: Being split into two, I am not sure that helps that much. Because I was saying like, I'd like to put them there.
>
> Modeller R: Yeah.
>
> Modeller R: Yea, that worked really well with the jugs, cause I was able to do the adding or changing easily, which was just come back together. But not Sudoku … [laughing] … No.
>
> Modeller S: It's such a high degree of integration with all those things, isn't it …
>
> Modeller R: Yea.
>
> Modeller S: it makes it really difficult to do this.

Although there might be many possible partitioning strategies for solving the Sudoku puzzle whereby each person takes partial responsibility for solving the puzzle (either based on digits or regions on the grid, for instance), there is no doubt that the underlying dependencies between squares are complicated. To some extent, the coordination problem in collaborative Sudoku modelling highlights the difficulties in supporting highly-dependent collaborative modelling. While one may argue that such situations are better dealt with individually, the collaborative Sudoku modelling shed some light on how well an EM approach might work for difficult-to-coordinate problem situations.

6.4 Case study IV: the cricket project[129]

This section discusses the cricket project (Beynon, 1993). The primary focus of this case study is how the teams practised an EM approach without the recently developed collaboration tool support. Whereas the previous case studies looked closely at the interaction between modellers, the case study of the cricket project examines collaboration through an EM approach at the project level. While the coursework was aimed at practising EM in the early stages of a systems development life-cycle, this case study seeks to explain the significance of integrating different perspectives contributed by individuals within a systems development context. Therefore, the aim of the study is to determine how the student modellers made the EM model for cricket simulation through integrating different perspectives within a group.

6.4.1 The background of the project

The cricket project (Beynon, 1993) was a second year undergraduate software development project. It was the main piece of assessed coursework in the "Introduction to Software Engineering" module. The students were asked to build a cricket simulation system for a fictitious county cricket club using an agent-oriented approach (i.e. EM). There were 122 students in the class. The students were allocated into 11 groups of 11-person teams[130] based on their knowledge and skills in 9 areas[131]: i) Unix familiarity; ii) workstation experience; iii) programming skills; iv) mathematical ability; v) oral communication skills; vi) technical writing ability; vii) leadership qualities; viii) enthusiasm for the project; ix) knowledge of cricket. To begin with, the students were given a booklet on the cricket rules

[129] This case study is based on the presentation "The co-evolution of system and developer's understanding in practicing an Empirical Modelling approach to systems development: a case study on cricket simulation" in WPCCS 2008 (Chan 2008), and on the technical research report CS-RR-444 (Beynon and Chan 2009), with new ideas introduced. In particular, table 6.16, and figure 6.17 and 6.18 appeared in the presentation slides of WPCCS 2008 (Chan 2008), and table 6.19 appeared in the research report CS-RR-444.

[130] Team 9 was an exception, which had 12 students. In addition, there were visiting students in both team 3 and team 6, one in each team. The choice of an 11-person team was more or less influenced by the constitution of a cricket team. The purpose behind this arrangement was to make it feasible for each member of the team to play the role of a player agent within a cricket game and make internal observations as described in (Adzhiev et al. 1994a, Sun 1999).

[131] The students were asked to rate their knowledge and skills in 9 areas in a survey prior to project. The allocation, therefore, was based on the results of the survey.

(MCC, 1990), a sample score sheet, a sample match record and a simple EM model of a cricket match. The project lasted 10 weeks during the spring term in the academic year 1993/94. Each team had a third-year undergraduate student as tutor for the project. The tutors provided consultancy to the students and monitored their progress throughout the project. The students also had access to UNIX workstations, a version control management system (viz. SCCS), and the principal EM tool support – tkeden. However, there was no real-time collaboration tool support, as the distributed version of tkeden had not been developed. This configuration reveals how EM might be carried out in a larger team (compared to the teams in other case studies in this chapter) with asynchronous collaboration tool support. It also gives clues as to what might be needed in the tool support for asynchronous collaborative modelling.

Technically, the simple cricket model is made up out of *agents* and *observables*, and they are connected by *dependencies* (cf. §4.1). The significance of using dependencies in place of listeners has been discussed elsewhere (Heron, 2002). In some aspects, the modelling paradigm used in this modelling exercise is similar to event-driven modelling, in that agents are made to respond to changes in observables (cf. §4.3).

The simple cricket model ("urchin cricket") was constructed from a commentator perspective on a primitive form of cricket match, that is, as a SMMA model (in the sense as described in section 5.1.2). The course of a cricket simulation (when using the model) is, by and large, driven by human interaction with the model. That is to say, the modeller drives the simulation through playing the role of the agents as described in section 4.1. This is significantly different from most computer simulations, which often take parameters at the beginning, compute a sequence of scenes, and visualise or render the resulting computation. This simple model has various purposes in this project:

i) To serve as a means of developing a requirement specification through animation;

ii) To be an artifact for the students to experiment with in deepening their understanding;

iii) To be a basic prototype for the students to begin with.

The main tasks for each team were to carry out activities that are predominantly in the early stages of systems development, e.g. requirement elicitation, understanding the problem domain, and project planning and management. They were also asked to generate an agent-oriented specification of the cricket simulation and to construct a detailed model of cricket based on the given simple cricket model using an EM approach. Although the focus of the exercise was on the early stages rather than on a full lifecycle of systems development, each team still had to develop a quality assurance strategy to assure the quality of the artifacts they were producing. This included carrying out review meetings and the use of quality checklists (as in a typical systems development project).

During the course of the development, the students were encouraged to enrich the model through observation of a cricket game as it happens in the real world taking account of many different *modes of observation* (Adzhiev et al., 1994a; Ness et al., 1994). The project was not aimed at full-lifecycle software development; the primary focus of the project was on the activities in the early stage of the development. To some extent, the context of the cricket project resembles the preparation of a tender bid for a software project in the public domain.

6.4.2 Method of study and approach to data analysis

Since the project was carried out more than 15 years ago, it would be difficult, if not impossible, for the participants to recall their experience of the project through interviews or questionnaires. Consequently, the analysis in this case study was primarily document based. The analysis has consulted the following documents:

i) *Preliminary Team Report* (PTR) – The PTR was the first report that the students had to submit in the project. In most cases, it consists of a feasibility study of the problem, a project plan, the individual responsibilities, and the quality assurance mechanism that the team has adopted.

ii) *Preliminary Requirement Analysis Report* (PRA) – The PRA is most appropriately viewed as an interim report of the project. There were 2 versions of the PRA. The individual PRA usually contains the detailed cricket model and its rationale. The teams' PRAs are similar to the individual PRAs, but the perspective on the model was collective and integrated the common and good

features from the individuals' models.

iii) *Final Team Report* (FTR) – The FTR was the last deliverable and the concluding report of the project. Most FTRs had documentation in the form of a project schedule, minutes, a preliminary design, an implementation plan, and the definitive scripts of the team's EM model of the cricket simulation.

iv) *Project Handouts* – These included the project guidance, the exercise sheets, and a couple of other documents that were given out during the project (Beynon, 1993).

v) Archives of emails and newsgroup discussions between the students and the module organiser.

Although the analysis in this case study was primarily based on documents, this should not have much impact on the quality of the case study. It is believed that the collaboration process was largely document-centric. This is to say, the collaboration process might be well captured and described, either directly or indirectly, by the documents such as project reports, meeting minutes, and the definitive scripts that they have generated in the process. This is because real-time collaboration EM tool support (e.g. dtkeden) was not available and therefore the students were forced to collaborate through other means such as face-to-face subgroup meetings (which in turn were reported in meeting minutes).

In order to develop a preliminary understanding of how the project was managed and how the teams approached to the project, I performed a preliminary analysis on the project handouts that were given out by the module organiser during the project and the archives of emails and newsgroup discussions between the students and the module organiser. The preliminary analysis did not provide conclusive results but provided essential information that was found useful in the later stage of the case study and has contributed to the discussion of the background of the project (cf. §6.4.1) and the discussion of this case study (cf. §6.4.3 and §6.4.4). These included:

i. Important milestones (i.e. dates) within the project.
ii. The composition and expertise distribution of the teams.

iii. What was the expected outcome of the project (e.g. what was the intended scope of the development process).

iv. The context of the development process (e.g. what tools were available and what were not)

Since the exploratory model construction process is the centre of the development process when practising an EM approach, the cricket simulation models become the primary object for the study analysis in this case study and the project reports are used as a supplementary resource. By glancing through all the team's reports, individual's reports, and the team's cricket simulation model that is embedded in the team's report, I found that most teams had come up with similar observations and similar structure within their models. They more or less followed Beynon's suggestions (Beynon, 1993) when practising EM and using tkeden to construct their models. However, larger variations were detected when I compared the individual models against each other and against the corresponding team's model. This motivated us to have a longitudinal study of the development process within one chosen team to study how they collaborated throughout the project in detail. By studying both individuals' and teams' models, various aspects of collaborative work that exploits an EM approach can be revealed, such as how their knowledge and personal interest influenced task allocation, their roles and the quality of their observations, how the team took advantage of diverse perspectives, and how the team resolved conflicting perspectives. This also helps us to understand the potential of an EM approach for collaborative work such as that involved in the early stages of systems development.

In the next subsection, I discuss in more detail how the analysis on the cricket simulation model construction for a particular team was carried out, and in particular, how their definitive scripts for the cricket model were analysed.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
Cricket Expert	-	-	-	-	Y	Y	-	-	-	Y	Y
Systems Analysis	-	Y	-	Y	-	-	Y	-	Y	-	-
Non-cricketer	-	-	Y	Y	-	-	Y	Y	Y	-	-
Technical Expert	-	-	Y	-	-	-	-	Y	Y	-	-
General Programmer	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Background Research	Y	-	Y	-	Y	V	Y	-	-	Y	-

Table 6.16 – The distribution of expertise in team 11 (extracted from the minutes attached in Team's 11 FTR).

6.4.3 Team 11's cricket simulation project

After the preliminary investigation of the team's model, I randomly picked a team – team 11 – for the detailed investigation of how the team collaborated throughout the project. According to their FTR, team 11 started their project with an introductory meeting on 19 January 1993. As in many commercial projects, the first task they decided to do was to find out their expertise and their area of interests in relation to the direction for investigation of the cricket simulation. Table 6.16 shows the distribution of their expertise. In a subsequent meeting, the team divided into four sub-groups to study aspects of the problem, such as "how do we simulate the cricket game?", and "how is the game actually played?" For this reason, each sub-group had been assigned a cricket expert for sub-group coaching and learning about the situation to be modelled.

After some brief study of the problem context, the team re-formed the sub-groups in relation to particular areas of concern in the cricket game – there were sub-groups for non-technical issues relating to cricket as the subject domain such as observing i) the bowlers and the umpires, ii) batsmen, iii) fielders, and sub-groups for technical issues such as score-keeping[132] and graphical simulation. As the deadline for submitting preliminary reports approached, the team re-formed again and focused on the technical issues rather than the modelling of the cricket game (what they regarding as the "control issue").

[132] Score-keeping can be a technical issue, and a non-technical issue. On the one hand, when it is treated as a technical issue, how the agent that presents the 'data' in a tabular format as in the scoreboard or a score-sheet is 'implemented' becomes significant. On the other hand, when it is viewed as a non-technical issue, determining what real-world agent awards the score and analysing the observational protocol they follow become a significant feature in the modelling context.

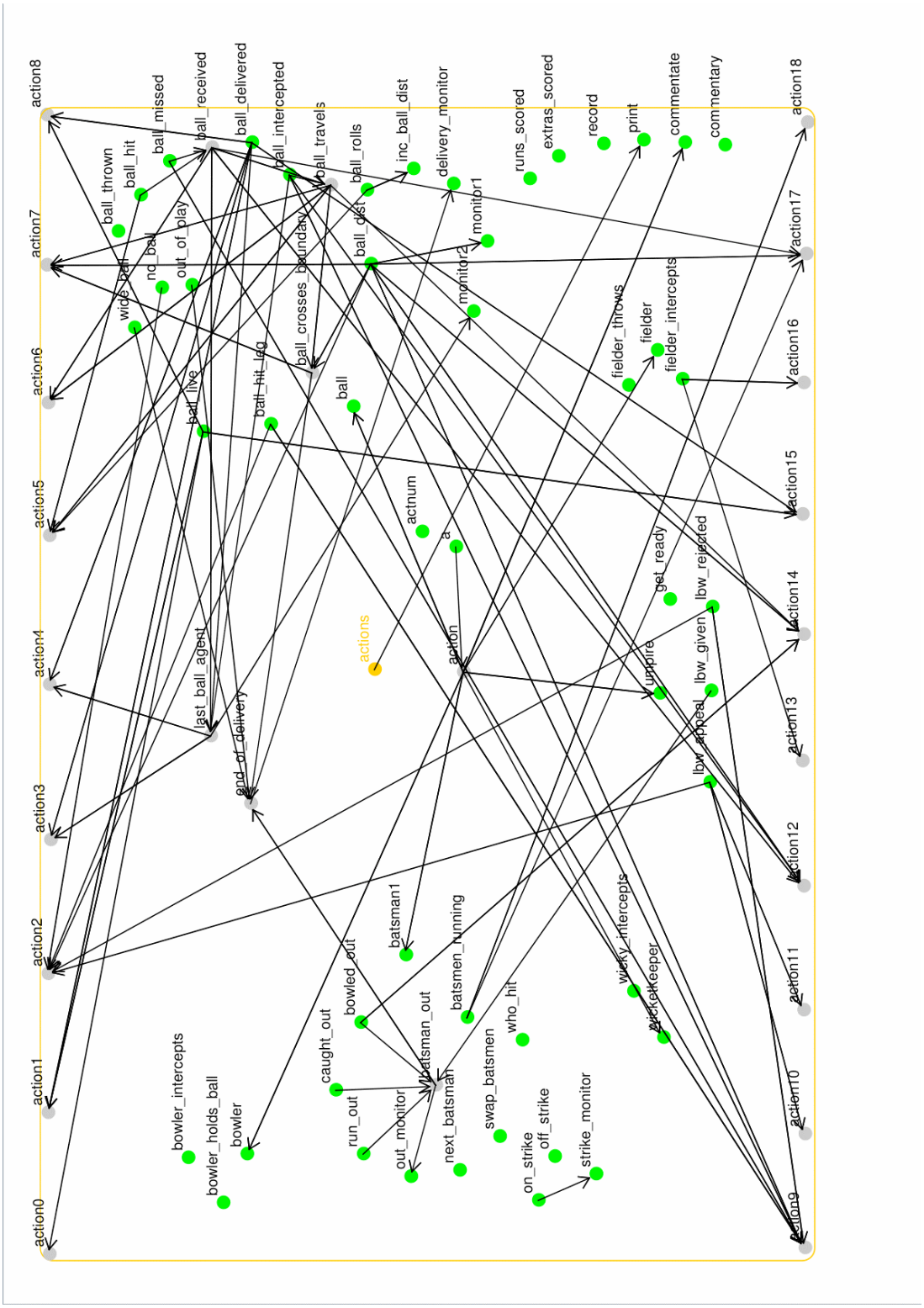

Figure 6.17 – The dependency graph for the team model of Team 11[133]

[133] The dependency graph is plotted using the Dependency Maintenance Tool (DMT) developed by Wong (2003). The green dots correspond to the observables in the model, and the grey dots correspond to the actions from which

To understand how the agents are connected within the team's model, I have plotted a dependency graph (cf. figure 6.17) based on the key observables and agents in the model using the Dependency Maintenance Tool (DMT) developed by Wong (2003). In the dependency graph, relevant observables were placed close together. The green dots correspond to the observables in the model, and the grey dots correspond to the actions from which a commentator agent can select. The orange border corresponds to the current scope of the model as defined by the total set of observables and agent actions currently modelled. The 19 actions (viz. action0 – action18) which are located on the edge of the orange boundary are the most important "key observables". These correspond to the definitions that shape how the cricket simulation can be animated. In EM terms, these definitions in fact manifest *dependencies*. For instance, action0 is defined as:

```
action0 is [ "Umpire says ready for play", 0,!ball_live ];
```

which denotes that action0 can be performed when the ball is not live. The dependency graph has provided crucial clues for understanding how a cricket game might be simulated through the team's model. The dependency graph analysis has also revealed that the team has pretty much followed Beynon's (1993) suggestion (cf. §6.4.1) of observing, modelling, and simulating a cricket game from a commentator perspective (or as an external observer of a cricket game). During the simulation of a cricket game with the EM model, the modeller plays the role of the commentator agent in the model. In order to progress the simulation, the modeller would have to make state changes to the observable "action" (as depicted in the middle of figure 6.17, below the observable "actions" in orange colour). Note that it would be misleading to think of the cricket model as a "product" in the sense of traditional systems development. At no stage is there a preconceived fixed interaction pattern with the EM model (cf. chapter 4). As I discuss in the next section, the modellers of the cricket project not only co-constructed the simulation, but also a shared understanding of the game.

---

a commentator agent can select. There are 19 actions in total (0-19) which are located on the circumference of the diagram. The orange border corresponds to the current scope of the model as defined by the total set of observables and agent actions currently modelled.

As I discussed in chapter 5, one of the key concerns in collaborative modelling (at the collaboration degree of engagement) is to facilitate diverse perspectives. Consequently, to answer the question "how did the team collaborate?" (the aim of this case study), we need to know how they facilitated diverse perspectives. To some extent, individual perspectives on the cricket simulation are embedded in the individual models (in the sense of EM construals; cf. chapter 4). Therefore, a more practical question would be "how did they negotiate the team's model as an integration of individual models?" However, apart from the work break down captured in the minutes, there was no discussion about how the team integrated individual models – it remains a mystery in the project documentation. For this reason, I analysed the differences among the team's model and the individuals' models. This analysis was informed by the grounded theory (Strauss, 1987; Strauss and Corbin, 1998). The coding strategy used in this analysis was similar to the one that is advocated in grounded theory. During the initial coding stage (i.e. open coding), the clustering in the dependency graph (cf. figure 6.17) was used to inform (but not restrict) the coding. Observables and definitions in the team's model that can be derived from the observations of different aspects of a cricket game were identified. For instance, the observables *'lbw_appeal'*, *'lbw_given'*, and *'lbw_rejected'* can be classified as "Leg Before Wicket (LBW) and its appeal". Likewise, the definitions

```
action2 is [ "Fielder appeals for LBW" , 2, !lbw_appeal && ball_dist < 1
&& ball_hit_leg && !no_ball && !lbw_rejected];
action10 is [ "Umpire says batsman out for LBW", 10, lbw_appeal ];
action11 is [ "Umpire says batsman not out for LBW" , 11, lbw_appeal ];
```

fall into the same category. After the coding of the team's model, I used the same coding scheme for the individuals' models. However, the original coding scheme does not reflect the subtle differences between the individuals' models and the team's model. In the later stage of the coding (i.e. axial coding), the coding categories co-evolved with the analysis as I "played" with the data further (cf. Yin, 2009). For instance, the LBW was later split into two categories, namely, "considered LBW", and "considered LBW and appeal to umpire". The vital difference between these two categories is that the former merely captured the act of giving the batsmen out LBW, and the latter captured the LBW appeal process. This difference also highlights the quality of observation that is captured in the model. For a

model to capture the LBW appeal process, the model must have captured the LBW situation. Therefore, I reviewed the observables and definitions that were codified in each category. Thirty-nine (39) categories were grouped into thirteen (13) scenarios. The final coding is depicted in figure 6.18. The definitive scripts of the teams' model (on the leftmost in the figure) and the individual models were listed side by side during the coding. Different parts (key observables and definitions in the definitive scripts) of the models were tagged in different colours according to the categories in the evolving coding scheme.

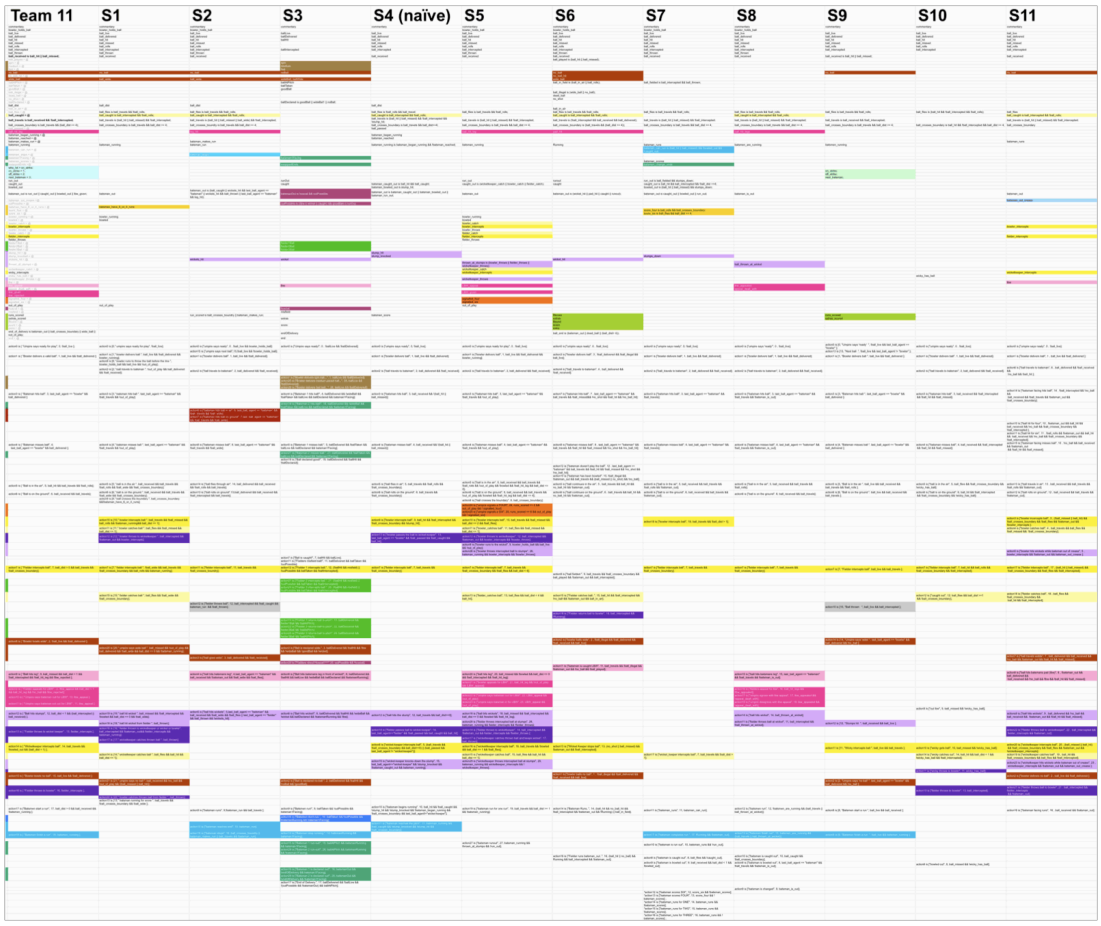

Figure 6.18 – Colour coded team's model and individuals' models listed side by side[134]

[134] The text (i.e. definitions in the models) in this diagram is not intended to be readable. The purpose of this diagram is to show how the analysis was carried out.

Scenarios	Team	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
a. How the ball is delivered	-	-	-	1	-	-	-	-	-	-	-	-
b. Abnormal deliveries	1	2*	/	2*	-	-	2	-	-	2*	-	1
c. How the ball is hit	-	-	1	-	-	-	-	-	-	-	-	-
d. How batsmen run after hitting or missing the ball	2	-	1	2	1	-	-	1	1	1	-	-
e. LBW and LBW appeals	2	-	1	1*	-	2	1	-	2	-	-	1
f. How batsmen are dismissed	-	-	-	1	-	-	-	-	-	-	-	-
g. How and by whom the wicket was broken	/	2	2	1	2	4	/	/	1	1	-	3
h. How the ball is intercepted	2	3	1	1	3	3	2	2	1	2	2	3
i. How the ball is caught	1d	3	1d	/	3d	3	2	2d	d	2p	/	3d
j. Modelling the transition between balls	1	3*	/	-	2	2*	1	-	-	/	2	2
k. Multiple batsmen and fielders	2	-	-	3	-	-	-	2	-	2	-	-
l. Scoring when the ball crosses boundary	1	-	1	-	1	1	1	2	/	1	/	-
m. Keeping scores	-	1	1	1	1	1	2	1	-	1	1	-

Table 6.19 – Contribution by the members in Team 11

After the coding, the next step in the analysis was to identify and to evaluate individual modeller's contribution to the team model. As shown in table 6.19, individual models were awarded with a numerical rating according to the quality of observation they captured in respect of each scenario (i.e. how much detail their model captured from the cricket simulation situation). These numerical ratings also reflect the degree of attention that individual modellers have given to those scenarios. Thus, a higher rating signifies a greater degree of attention (i.e. more details from the scenario have been captured), and '-' denotes that no attention was given to the scenario at all. For instance, the models that have merely captured the LBW situation have a rating of 1, while the models that have captured both the LBW situation and the LBW appeals would have a rating of 2 (cf. scenario (e) in table 6.19). In another example, '1' has been awarded to S2, S3, and S8 because their model only captured one possibility: the ball is retrieved by the fielders, while '2' has been awarded to the team, S6, S7, S9 and S10 because their model has captured two possibilities: retrieval by the fielders or by the wicketkeeper. In addition, the symbol '/' refers to a more passive approach in modelling the situation in which the fact that they had given attention to a scenario might almost go unnoticed. For example, in the team's model, no observable or

definitions were found to be directly related to how and by whom the wicket was broken (i.e.. scenario (g)). The definition:

```
action12 is [ "Ball hits stumps", 12, ball_dist < 1 && (ball_intercepted
|| ball_received) ];
```

models the scenario more passively (in terms of observable, dependency, agent and agencies in EM) when compare to S5's model, in which observables are linked up by dependencies:

```
thrown_at_stumps is (bowler_throws || fielder_throws ||
wicketkeeper_throws);
action26 is ["bowler throws intercepted ball to stumps", 26,
batsman_running && bowler_intercepts && !bowler_throws];
action27 is ["batsman runout", 27, batsman_running && thrown_at_stumps
&& !run_out];
action28 is ["fielder throws intercepted ball at stumps", 28,
batsman_running && fielder_intercepts && !fielder_throws];
action29 is ["wicketkeeper throws intercepted ball at stumps", 29,
batsman_running && wicketkeeper_intercepts && ! wicketkeeper_throws];
```

Moreover, the same scenario can be modelled in many different ways (e.g. by observables complemented with triggered actions, or solely by a set of dependencies). To distinguish different uses of definitions, letters are appended to the numerical ratings in table 6.19 – 'd' indicates that the scenario was modelled without the use of any procedural action, and 'p' indicates the use of procedural constructs. Unusual contributions are denoted with asterisks. For instance, in modelling the transition between one ball and the next, S1 not only considered the throwing of the cricket ball between cricket players (viz. fielder, wicketkeeper, and bowler), but also the catching of the cricket ball by the cricket players. By comparing the team's and individuals' model side-by-side, similarities between them can also be identified. Contributions were attributed to an individual model if the parts were close in semantics. For example, the observable *'ball_hit_legs'* introduced by S8 is viewed as significant, as the observable *'ball_hit_leg'* is a contribution to *'ball_hit_leg'* in the team's model in the aspect of the LBW appeal process (cf. scenario (e) in table 6.19). In most cases, significant individual contributions can easily be identified because they are taken into the team's model without modifications. In some scenarios, however, it is difficult to tell who is the main contributor – the team simply integrated the common perspective that was shared by the majority of individuals (e.g. scenario). In table 6.19, shaded squares mark the individual perspectives

that were adopted in the team's perspective.

This analysis seems to suggest that the team's model was not integrated by one person[135], but through negotiation of diverse perspectives embedded in the individual models. It also suggests that the collaboration between the students was close to *genuine participation* in the participatory development model that I described in section 3.4. In such a development process, participants gain insight about the artifact (i.e. the simulation of the cricket game with an EM model), as well as contributing freely with different perspectives and with different knowledge. In the next subsection, I discuss the results of this analysis in relation to the EM approach.

6.4.4 Limitations

There are at least three research limitations in the cricket case study. The first limitation comes from the data collection technique. As in the VEL case study (cf. §6.1), the cricket case study was mainly based on archived documents. While the documents are readily available to be reviewed and analysed, they can only be trusted to a limited extent due to the unknown bias of the documents' authors. This is to say, the documents might have been drafted to give an illusion that the model were developed in the way that they described (cf. Duffy, 2005). For instance, it could be the case that the documents were written to give an impression that the project was a collaborative effort when in fact it was not. However, it would be difficult for the students to perpetrate this deception due to the structure of the cricket project: the students had to submit both individual and team reports at various stages of the project (cf. §6.4.1).

The second limitation comes from the fact that only the final definitive scripts were available. That is, no intermediate definitive scripts were available for further analysis. Despite the fact that there are traces of the development process in their meeting minutes and individual reports, it is difficult to be sure that the definitive scripts of the cricket simulation models were not created at the last minute.

The third limitation comes from the coding techniques used in the analysis of the definitive

[135] For example, by a chief software architect in the traditional systems development.

script. There can be endless options for coding and comparing the same set of data (Flick 1998). Where the quality of observation in the model construction is concerned, it seems natural to code according to the aspects in a cricket match that the cricket model captures, rather than to the features and functions it implements as in a cricket simulation program. Certainly, this does not preclude the possibility of coding against features and functions in future studies.

6.4.5 Discussion

Neither the project documentation nor the models made this project interesting; it was the collaboration process behind the construction of these artifacts that made this case study interesting. The lack of synchronous tool support in the cricket project had forced the modellers to collaborate asynchronously most of the time, and this makes it stand out from the other case studies in this chapter. In the rest of this sub-section, the virtues of an EM approach to collaborative modelling are examined by drawing on the analysis of Team 11's cricket project, and with reference to figure 6.17, 6.18, and table 6.19.

Facilitating and integrating diverse perspectives

As Beynon pointed out to the students at the beginning of the cricket simulation project, EM should not be used as a hacking tool for 'quick and dirty' prototyping, but as an *instrument* (Beynon et al., 2000) to facilitate the observation that is the analogue of doing experiments in other scientific disciplines (Beynon and Russ, 2008) throughout the modelling process:

> *"Its [EM approach] focus is not upon 'hacking a program till it works', but upon thoughtfully correlating the behaviour of variables in the computer model with the real-world observations they represent. In spirit, this can be regarded as following the pattern of the experimental method in science, and should be subject to the same discipline." (Beynon, 1993)*

As I discussed in chapter 4, the EM artifacts are construals that are based on modeller's subjective and personal experience of the situation being observed. As was the case in team 11, different modellers (i.e. members of the team) adopted different modes of observation of the relevant situations (i.e. scenarios in a cricket game) at different times. For

instance, a modeller may begin with external observation of the cricket ball, and trace a sequence of states that follows the trajectory of the ball, to see what kind of state changes the ball may cause in other agents as in a cricket match in the real world. In other times, modellers may make observation from the perspective of an internal agent, e.g. to see what types of state changes the umpire (i.e. an agent) can make in the context of a cricket game.

Different modes of observation may result in different degrees of attention to the details within the context of the cricket game (i.e. the problem domain of the simulation), and that leads to diverse perspectives on the same problem domain. Our analysis of the team's individual models is in line with this claim. As shown in table 6.19, individuals have different ratings for different scenarios. This reflected the fact that different degrees of attention were given to different aspects in the problem domain, even by the same individual modeller.

Despite the potential they offer for a richer understanding of the problem domain, such diverse perspectives posed a challenge in integrating the individual models. The analysis of all teams' FTR seemed to indicate that, in most cases, the team chose to address this issue by selecting one or the other, instead of venturing to introduce dependencies to reconcile variants of the same event or situation. This is corroborated by the fact that numerous individual contributions have been discarded (e.g. scenario (b), (g), (h), (i), (j), (k), and (l) in table 6.19)[136], and contributions in some scenarios were entirely ignored (e.g. (a), (c), (f), (m)). It may be that the issues in the integration of diverse perspectives stem from potential conflicts of scenarios or subtle group dynamics, but this cannot be fully verified in either case. Despite these integration issues, allowing diverse perspectives potentially offers a richer understanding of the problem domain and a higher degree of individual engagement, and this makes collaboration closer to *genuine participation* (cf. §2.5.2 and §3.4).

Learning and co-construction of the problem situation

Our analysis seems to indicate that in some scenarios students with less sophisticated knowledge of cricket were the more successful in identifying the most appropriate way to

[136] For example, the team's model does not integrate all contributions from S6's individual model for scenario (b), and does not consider S5's or S11's modelling for scenario (g).

model the possible actions of agents in a cricket game, based purely on their observation and immediate experience. For example, S1 and S9 showed a higher degree of attention to abnormal deliveries of the cricket ball (i.e. scenario b), whilst S3 and S4 had the most detailed model for scenarios (d) and (h, i) respectively and these were subsequently taken into the team's model (cf. table 6.16 and table 6.19). One explanation is that the novice cricketer exploited the EM process for a learning purpose. This idea has been explored in *constructivist computing* (Harfield, 2008; Beynon, 2008). Another explanation is that the students with more experience of cricket found it harder to analyse their 'intuitive' knowledge but felt they would be able to retrieve this as necessary. For instance, as one such student said:

> *"I decided to give a minimilistic [sic.] simulation just to show I understand how to program [model] the requirements. The design can be developed quite easily into whichever level [is] wanted." (S11's PTA)*

As mentioned in chapter 5, *co-construction* in collaborative modelling involves the co-construction of shared understanding of the shared activity and of the problem situation. Hence, if we view the development of the simulation of a cricket game as a collaborative modelling process, the co-construction of the team's model would be the shared activity, and the co-construction of the understanding of how to simulate a cricket game would be the problem situation. In this sense, the learning through the construction of the individual cricket model seems to have played a crucial role in developing a shared understanding of the problem situation. It enables individual understandings of the problem situation (i.e. embedded in the individual models) to become a shared understanding (i.e. embedded in the team's model) through negotiating diverse perspectives into a unified shared perspective (i.e. the integration process).

6.5 Concluding Remarks[137]

Even though research into EM has progressed tremendously over the past two decades, there have been very limited studies and documentation of the EM process through research techniques that are common in social science experimental practice. In particular, there have been virtually no attempts at gathering data about modeller interaction in collaborative modelling with an EM approach through observational techniques such as video recording and participant observation. At present, there is insufficient expertise in EM available to carry out larger scale of observational studies in collaborative EM that involves many modellers. It would therefore be difficult at this stage to attempt rigorous experimental analysis of the practice of EM using the kind of experimental studies that we would expect in investigating a well-established practice.

The case studies presented in this chapter represented some of the very first attempts at ethnographical studies of the EM process. From a holistic perspective, these case studies have the following characteristics:

i. Relying on the analysis of the archived documentation of collaborative projects that had practised an EM approach without access to the participants in those projects (viz. the VEL case study discussed in §6.1 and the cricket project case study discussed in §6.3).

ii. Documentation of a collaborative modelling process that practised an EM approach from a first-person experience[138] (viz. the first distributed jugs discussed in §6.2).

iii. Investigating the distinctive characteristics of EM through multiple data sources. For instance, the examination of the interaction history in conjunction with other data sources, such as video footage, screen video capture, and archived documentation.

---

[137] This section highlights some backgrounds and limitations concerning the empirical study of collaborative modelling with an EM approach using observational research techniques that are common in social science experimental practice. This is a topical issue which is fruitful for further research into EM. The discussion in this section is a result of intensive discussion between myself and my thesis supervisor, Dr Meurig Beynon, which was based on the thesis examination committee's comments on this topic.

[138] Due to the fact that the modeller's experience is one of the primary focuses in EM (cf. Beynon 2009), it is appropriate to include the description of the engagement in EM from a first-person perspective, no matter how difficult it might be to reconcile this with the need for objectivity in the results of the case studies.

Apart from the collaborative projects presented in the case studies, many hundred students have practised EM over the last two decades. Their reported experience seems to indicate that modelling with definitive scripts in EM has some distinctive characteristics that requires more detailed investigation in future research projects. For instance:

1. The interaction history in EM is an unusual resource that offers insight into the progression of the model-construction process. As the analysis in the case studies have demonstrated, the interaction history can be a fruitful resource for the reconstruction and investigation of the entire modelling process. Such use of interaction history in reconstructing part of the modelling process of Care's planimeter modelling can be found in Harfield's (2009) doctoral thesis. In fact, the interaction history is not only useful for the researchers who are interested in the modelling process, but also for the modeller him/herself. This is due to the fact that the interaction history, to some extent, can be used to 'revisit' the *state-as-experienced* as encountered by the modeller stage-by-stage in the modelling process. This potentially allows the modeller to retro-fix certain problems in the model – a feature which is not typically supported in the conventional systems development paradigm.

2. The role-shifting behaviour discussed in the case studies is itself "matter-of-factly" exploited by many EM practitioners. The modes of interaction underlying "interacting as a developer" and "interaction as a user" are virtually the same. The sole difference between these two styles lies in the perspective that the modeller has adopted to support that interaction and its interpretation. For example, "interacting as a user" typically refers to the fact that the modeller, in contrast to "interacting as a developer", is presuming little knowledge and experience of the observables that can help to frame the meaning of redefinitions. This highlights some logistic issues which may benefit from further empirical studies. For instance, "in what context does this role-shifting behaviour occur?", "how consciously does this role-shifting behaviour occur?", "how does this role-shifting affect the collaborative modelling process when practising EM?"