CHAPTER SEVEN

Case Studies

This chapter provides the practical case studies to help to understand – through knowing-by-doing – the principles of EM and the issues discussed in the last two chapters, such as the EM approach for system development, participative process modelling and participative BPR. The purpose of these case studies, developed during the research, is to illustrate the ways in which EM can be applied to software development and to participative BPR which have already been described theoretically in chapter 5 and chapter 6.

7.1 Introduction

This chapter aims to exemplify the features of the EM approach discussed in previous chapters through practical research. There are two case studies included in this chapter.

The first case study (in section 7.2) illustrates an ISM for a digital watch model (or *Timepiece* artefacts). In this model, three artefacts represent three aspects of timepieces: a digital watch, an analogue clock and a statechart. This modelling case study is to demonstrate the EM process for system development which was described in chapter 5 and chapter 6. This digital watch model was mainly developed by Dr Beynon and Dr Cartwright in 1992 and 1994 respectively, and is still being improved and modified today in the EM group. It is helpful to refer to Figure 4.2 and Figure 5.3 to have a general view of how the EM concepts and modelling process are applied in this case study.

Another case study (in section 7.3) involves modelling a business process of a warehouse (the manual distribution within a warehouse or between different warehouses). This model was developed by this author during his research. The purpose of this modelling is to show how to apply the SPORE framework for cultivating requirements of the warehouse management system and how the participative process modelling can be achieved through the distributed environment provided by **dtkeden**. For this, several computer models are constructed as artefacts in a distributed modelling environment in order to shape the action of agents associated with the manual distribution process within the warehouse. These agents are interacting with the models at the client nodes and the window at the server model shows the current state of the manual redistribution process in the warehouse environment (cf. Figure 7.6). Modelling within the server node provides a potential opportunity for modelling the system or business process from the perspectives of an external observer. Individual participants in the modelling process can 'communicate' with each other and have the 'visualised' insights of other modellers through different modes of interactions provided by the distributed EM tools. Further details of the features of distributed EM (DEM) and these modes of interactions can be found in the research work and thesis by Sun (1999).

7.2 The Digital Watch

In this case study, we will use the artefacts of the digital watch to illustrate the characteristics of the EM approach in system development. There are three artefacts in Figure 7.1: the model of a digital watch and an analogue clock, and a statechart which represents the functionality of the digital watch display mechanism. The clock face is the artefact for communicating time and the statechart is the artefact to tell the user or designer how the display functions of the digital watch are affected by the pressing of watch buttons as well as the level of the battery energy. There are four buttons surrounding the digital watch at each corner which respond to the mouse clicks. These buttons are linked by definitions to the statechart so the current internal state of the watch can be shown by the boxes with solid lines (rather than dotted lines). In the analogue clock face, the hands of the clock are represented by a graphical line drawing. The positions of these hands are linked to the time on the digital watch. The movement of the

hands metaphorically represents the mechanical coupling of the hands of a real analogue clock, e.g. when the second hand moves, the minute hand and, in turn, the hour hand move with it.

7.2.1 The ISM for a Digital Watch

In constructing the ISM for a digital watch, the modeller was working in a situated manner and with reference mainly to Harel's statechart but also having a real watch (the referent) in mind. The relevant observables for the ISM include the characteristics of the actual watch which can be reliably identified, through an interaction with the watch during a period of time. Examples of observables may include the appearance of the watch, the digits appearing on the LCD display, the layout of buttons, the presence and absence of the power supply, and the current mode of display. Of course there is no constraint on

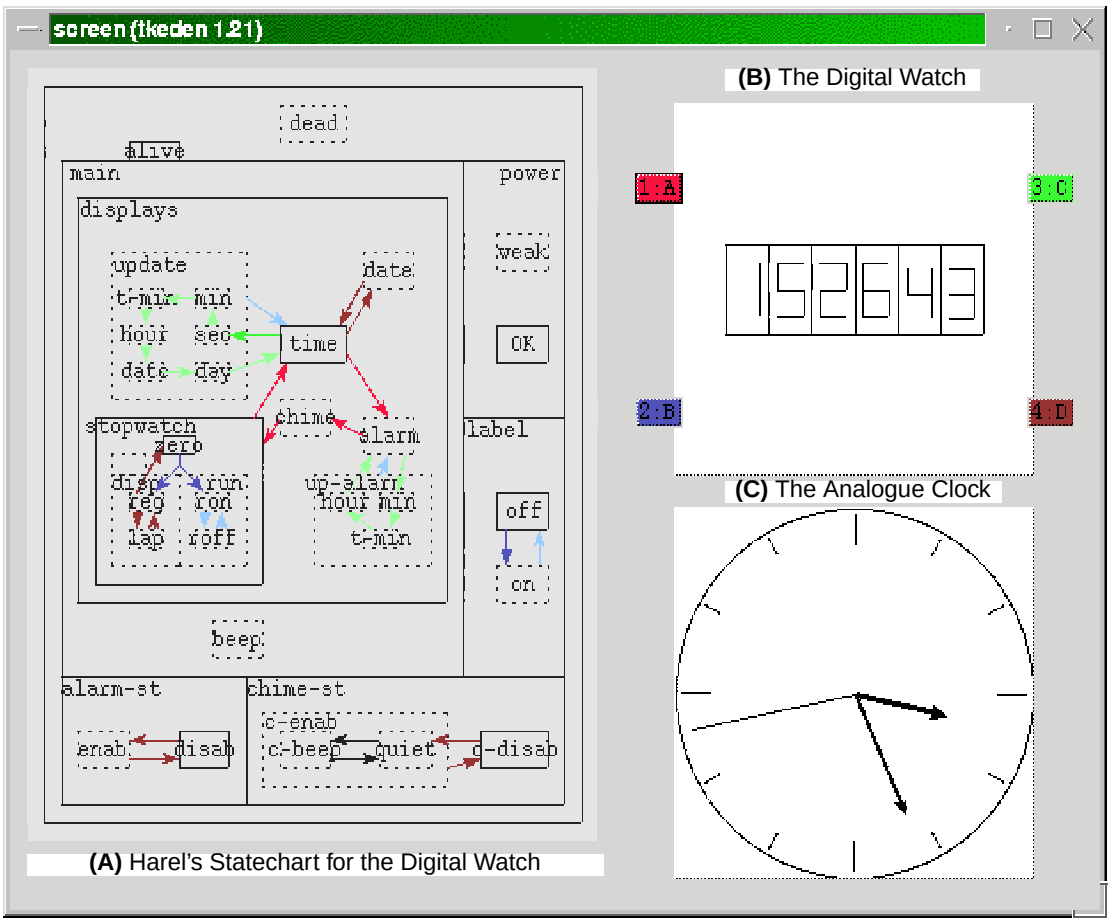

Figure 7.1 The Digital Watch

the choice of observables to suit a particular goal, and new observables can be added to the model when necessary. Adding new observables to the model can, as Beynon et al. (1998) suggest, "be viewed as recruiting more observers in the spirit of 'subject-oriented' programming". The only constraint may be how far the conception of the watch in the minds of new observers can be consistent with that of the original modeller.

The ISM for a digital watch metaphorically represents the observables of the actual watch (its referent). This model only depicts the rectangular shape of the watch, with LCD display and four buttons at each corner. There is also an analogue clock synchronised to the digital watch. The statechart expresses the significance of the current display by highlighting a particular region. The statechart was originally from Harel (1992) for the software development of reactive systems. The statechart in this case study is serving as a cognitive artefact (a term introduced in Norman (1991)) because some aspects of the state of the actual watch shown in the statechart cannot be directly observed by the modeller.

Different types of agency in the Digital Watch model are shaped by spreadsheet-like dependencies. Some of these dependencies link the visual elements with the invisible internal state of the model so they reflect the interaction of the modeller (the external observer) with the model. For example, the modeller can simulate the user's behaviour of button pressing through his interaction within the simple interface. He can extend or modify the definitions, functions and triggered actions from the input window at any time during the development. Other dependencies in this ISM include: changing the current time will affect the displays on LCD as well as the analogue clock; moving the minute hand of the analogue clock will move the hour hand; changing the mode of display will affect the highlighted state in the statechart and the display in the digital watch.

7.2.2 The EM Development of the Digital Watch Model

The association between the artefact and its referent is made in an experimental manner. The patterns of state changes in observables are identified through the modeller's observation and experiment, and

these identified patterns are subject to revision after subsequent observation. Referring to Figure 5.3 (the unified development procedure in EM), the process of developing an artefact involves incremental construction of a model which is guided by the evolving knowledge of the modeller (through his experience and experiment during the process) about the properties and situation of the referent. In this process, it is essential that the modeller has an insight of the current status of this model, and can revise this to consider new information and perspectives on the referent.

The simulation of this digital watch was firstly developed by Dr Beynon in November 1992. During that development the model was incrementally constructed being given intermittent attention while running as a background process on a workstation (Beynon and Cartwright, 1995). Based on this original model (the seed-ISM), the subsequent development process typically involved the addition of groups of definitions to add new visual components to the model (or redefining the existing definitions to modify them) as well as the addition of procedures (i.e. actions) in EDEN to simulate the newly introduced agents. When the seed-ISM has been constructed, the tasks of adding new components or behaviours are usually done by developing portions of the definitive script or tracing problems by extracting pieces of the script and testing them in isolation. For example, the statechart in Figure 7.1 (a) was firstly developed by describing the disposition and interaction of boxes abstractly without giving coordinate information. This coordinate information was then added by imposing the grid from Harel's statechart. The process of tracing abnormal or unpredicted behaviour of the model (i.e. the test and experiments arrow in Figure 5.3) was assisted by the close correspondence between the variables in the model and observables of the watch (the referent). The modeller's experiment with both the model and its referent can help to determine whether the simulation meets its intended behaviour.

The above work can be regarded as phase 1 of the development procedure of a digital watch. The further details of the statechart, and the button interface constituted phase 2 and were added by Richard Cartwright in December 1994. The model after phase 1 was 'frozen' and became a 'system' because it had satisfied what we needed at the time[1]. But at later stages the functionalities of the system may not be appropriate or not enough for the needs of users due to the change of the environment

or technologies. For example, in the model of an analogue clock in phase 1, the motions of both the second hand and the minute hand are defined independently via the `gettime()` function (but are synchronised by updating the position of the minute hand at regular intervals):

```
%eden
tn = gettime();          /* Get the current real-world time from terminal */
tnsecs is abs_time(tn);  /* The variable tnsecs records the number of seconds elapsed in real-time
                            from an arbitrary initial time */
t is tnsecs / 60;        /* The variable t representing time elapsed in minutes */
%donald
within clock {
    secAngle = (pi div 2.0) - float (tnsecs % 60)* (pi div 30.0)
                         /* "tnsecs % 60" determine how many seconds elapsed since last whole minute
                            time */
    minAngle = (pi div 2.0) - float (t % 60) * (pi div 30.0)
    hourAngle = (pi div 2.0) - float (t % 720) * (pi div 360.0)
    secHand = [center + {size_secHand * radius @ secAngle}, center]
    minHand = [center + {size_minHand * radius @ minAngle}, center]
    hourHand = [center + {size_hourHand * radius @ hourAngle}, center]
}
```

This might not cause a big inconvenience for the use in our everyday life if this analogue clock is simply used for interpreting the time. But we may need to improve this situation for a more accurate use in other environments. In 1994 Cartwright enhanced the model to include a chess clock which requires greater accuracy in the movement of the minute hand. This was achieved by binding the motions of the second and minute hands together indivisibly thus:

```
%eden
tn = gettime();
tnsecs is abs_time(tn);
%donald
within clock {
    secAngle = (pi div 2.0) - float (tnsecs % 60) * (pi div 30.0)
    minAngle = (secAngle / 2 * pi) * (pi / 6)
    hourAngle = (minAngle / 2 * pi) * (pi / 6)
    secHand = [center + {size_secHand * radius @ secAngle}, center]
```

1. For example, this watch artefact was ‘frozen’ at this stage as it was sufficient for the demonstration of EM principles to MSc students, i.e. how the characteristics of the statechart can relate to the presence of LSD agents and the roles they play.

```
        minHand = [center + {size_minHand * radius @ minAngle}, center]
        hourHand = [center + {size_hourHand * radius @ hourAngle}, center]
    }
```

This modification makes the clock reflect more closely the real-world referent in which the minute and hour hands are mechanically coupled with the second hand. The principle of a traditional spreadsheet was also added to the model at this stage to relate the power consumption of the LCD display to the pattern of use.

Following Beynon and Cartwright, two other EM modellers have contributed to building the watch artefact. The functionality of the watch was completed by Carlos Fischer in 1999, which we regard as the phase 3 of the development. He added new definitions and procedural actions to the model, and buttons for updating time and date and for switching the clock on through the interface. This resembles the design and test/experiments activities (i.e. the internal view of the model) in Figure 5.3. But during the same time the interaction with the evolving watch artefact (i.e. the external view) can also be achieved which, as described by the modeller (cf. Fischer and Beynon, 2001), played a far more significant role in the modelling activity than communication with the contributors. He also commented that the model itself is a more useful repository of knowledge about its construction than the memories of previous modellers.

The development of the watch artefact has been continued further by Chris Roe in this group which can be viewed as phase 4. Roe has modified the analogue and digital visualisation of the watch with buttons which correspond to the ones on his real watch (a sports watch). In this phase, a new visualisation of a mental model, based on the statechart in Figure 7.1 (a), has also been developed which the user may construct when learning to use the watch artefact (cf. Figure 7.2). It shows the state transitions which are familiar to the user and, as the user gains experience of interacting with the watch, the number of familiar states increases and they are added to the visualisation during the interaction process[2]. For example, through Figure 7.2 we can deduce that the user has explored changing the time of

2. The further details of the new digital watch with visualisation can be found in (Roe et al., 2001).

the main clock and altering the alarm, but not yet explored the other features. The reason for developing the new visualisation instead of the statechart is that though the statechart is a good mental model of a watch that users already understand – as it represents a complete view of a situation – it is not a good mental model of a watch when users understand only partially. The new visualisation in Figure 7.2 aims to represent the knowledge the user has of the watch at the present time. In this phase Roe also added the artefact of two runners who are completing a race and the aim is to use the watch to record their times when they complete the race. The visualisation of the runner artefact takes a very simple form: the progress of each runner is depicted by a dynamically extending line.

The development history of the digital watch illustrates the unified development procedure of EM (Figure 5.3) in which the EM model is always open to revision and extension. These modifications also in some ways indicate the potential for reuse in EM.

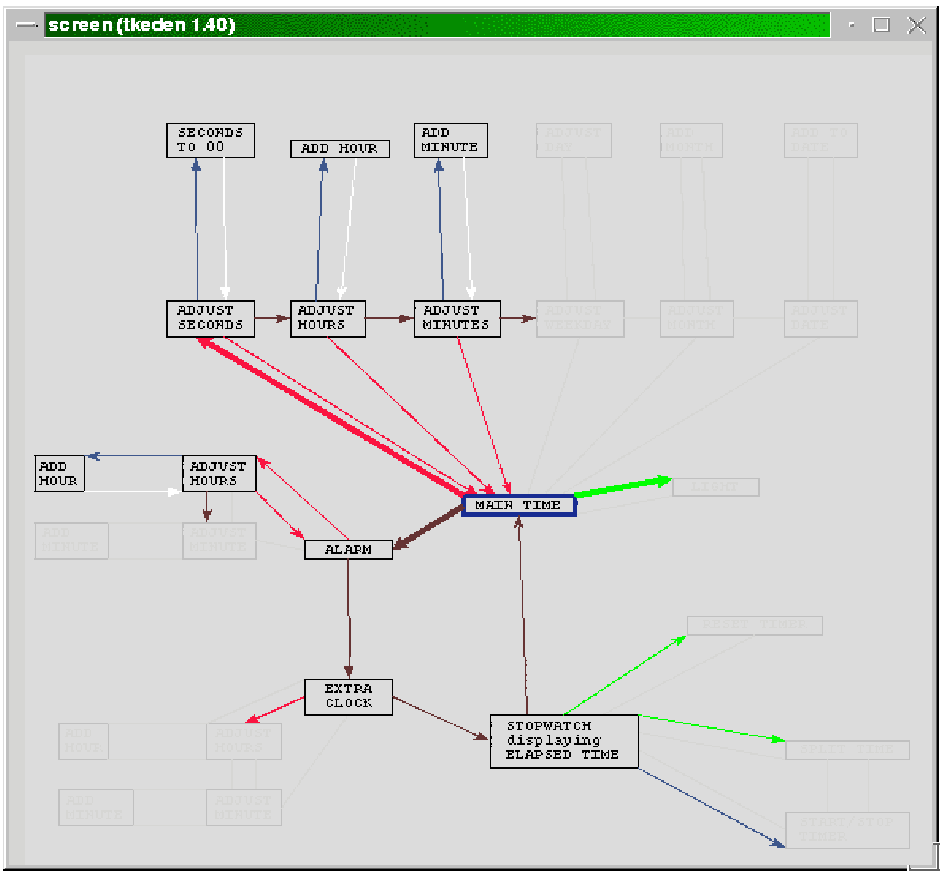

Figure 7.2 The Visualisation of Mental Model of the Digital Watch

Participative Process Modelling for the Digital Watch Model

The choice of observables for the digital watch is subjective and reflects the knowledge and perception of the human agents who interact with it. For example, interpreting aspects of the appearance of a watch can be subjective and will sometimes depend on different environments. The aim of EM is to integrate these different perspectives of the human agents as well as the automated agents within the modelling process. The distributed EM tools (**dtkeden**) can be used for this purpose. Several models can be run concurrently in the **dtkeden** configuration, and each of these models provides an artefact for the human participants to interact with. The synchronisation between the evolution of the models and the individual insights can visualise the participants' viewpoints and show them to each other. It also provides a channel of communication between these participants interacting with their own artefact. This distributed configuration of **dtkeden** can also serve as an environment for learning how to use a digital watch. For example, under the *private model* of interaction, the teacher (at the server model) can monitor the learning situation of each learner (at the client model). The teacher can give instruction on the functionality of the watch, see the way in which the learner's actions affect the statechart, and sometimes introduce the statechart into the learner's ISM when the learner needs the relevant knowledge about the functionality of the watch. This resembles the actual learning process because recognising the modes of display of a digital watch requires knowledge of the range of the watch functions. Familiarity with these functions require the user's experience about the association between button pressing and the functionality of the watch.

7.3 The Warehouse Management System

In this section, a warehouse management system is taken as a case study to illustrate the potential for applying the SPORE framework in participative BPR. This case study was adopted from the text *Object-Orientation Software Engineering (OOSE)* by Jacobson et al. (1992). The main concern of Jacobson et al. is to identify the requirements of proposed computer systems by analysis and modelling based on use case concepts. The main idea behind their use case approach is that if we understand the roles of

the users who will have access to the proposed system, then we can identify some of the essentials of the system usage and from which the requirements can be elicited. As described in chapter 5, each use case is associated with a particular kind of interaction between actors and the system and as such the interaction might be directed towards one of the required functions of the system, for example, the use case of manual redistribution between warehouses in the case study. In their other book *The Object Advantage* (Jacobson et al., 1995), they extend the use case approach to modelling business processes by introducing the concept of 'business use cases', and propose a method for object-oriented business engineering (OOBE). Here the use case model serves as a process model of the existing business (the external view of the organisation), which is used as the basis for prioritising the processes to be reengineered (cf. subsection 5.2.3).

As mentioned in previous chapters, we regard it as important to address BPR within a broader context of developing a business model. This means widening our focus to include the real world (i.e. the environment and human factors) rather than the computer system alone. When adopting this perspective, the role of EM is to develop a computer-based model which can be used to explore all the charac-

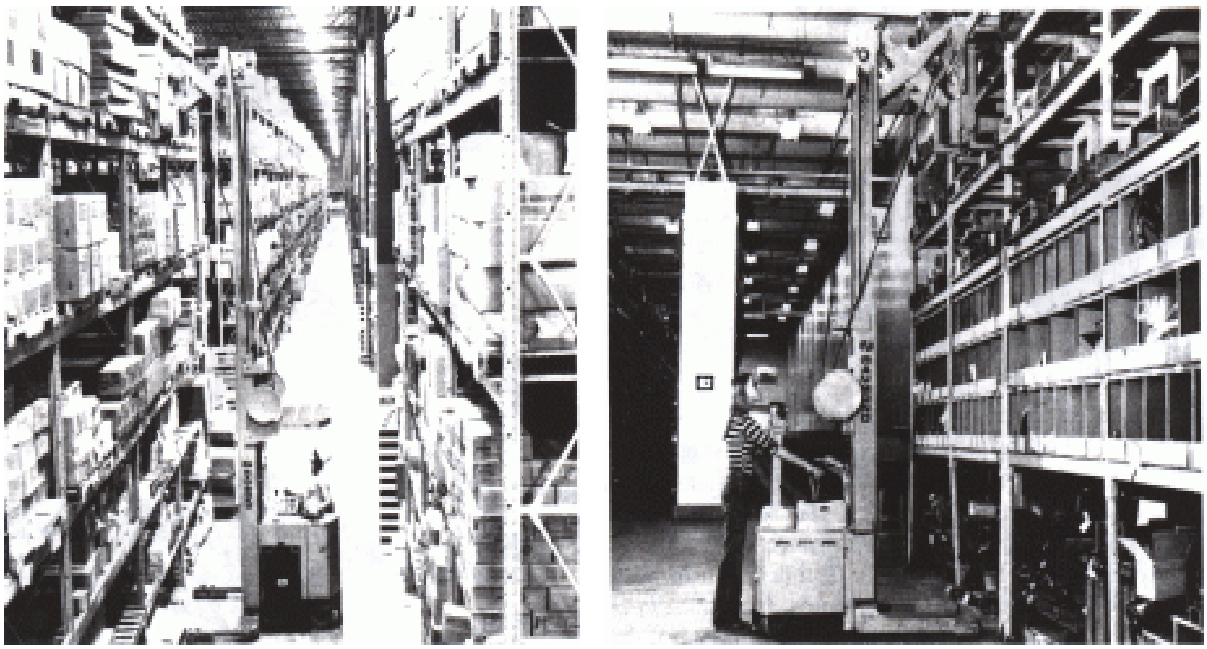

Figure 7.3 The Warehouse (from Magad and Amos, 1995)

teristic transactions of the warehouse. The character of our framework is through-and-through agent-oriented, so that the warehouse activity is conceived with reference to the state-changing protocols for both the human agents and automated components within the system. Because the human actions are constrained by the business process and follow some reliable patterns, it is possible to regard their cooperative activity as a form of computation (in the same way as we might say "the users are programmed"). The human users and the computer-based components can then be viewed as computational agents in a complex system. This agency is mediated through the user-computer interface: the input of the user influences the state of the computer, and the output of the computer changes the environment of the user (Beynon and Russ, 1994).

7.3.1 Introduction to the Warehouse Example

This case study involves applying the SPORE framework to a warehouse management system to achieve BPR. Our goal is to illustrate the use of the EM concepts mentioned in chapter 4. However it is not possible to give complete details and fully illustrate the entire study because the case of a real warehouse is too large[3].

The proposed system is to support warehouse management. The main function of a warehouse is to provide its customers with space to store goods for varying amounts of time, during the journey between the plants of production and the retail outlets; or to support companies that need local warehouses but may not have local offices (cf. Figure 7.3). The operations of the warehouse also include storing different kinds of items and using trucks (or any other forms of transportation such as railway) for the shipments and redistribution of items[4] (Figure 7.4). The aim of introducing computer systems into the warehouse is to give automatic support to the storage and redistribution. These processes involve keeping track of the locations and status of items, differentiating between kinds of items (for example

3. Jacobson et al. (1992) describe in their warehouse case study that the complete documentation alone would cover more pages than their text book (which is more than 500 pages).
4. Sometimes the term 'distribution centre' is synonymous with warehouse since most goods in a warehouse are in somebody's distribution system (Johnson and Wood, 1996). Thus in the distribution channels, warehouses are intermediate storage points between the manufacturer and the retailers.

they may be perishable or flammable, or some items must not come to contact with other items[5]), maintaining security and integrity checks, and managing storage, retrieval and relocation[6]. The people in the warehouse who will use this system may include: the *foreman* responsible for that warehouse; the *warehouse worker* who is responsible for loading and unloading; the *forklift operator* who drives a forklift in the warehouse (Figure 7.5); the *truck driver* who drives a truck between different warehouses; the *office personnel* who receive orders and requests from customers, arrange the truck routines, and keep records of all warehouses; and the *customers* who own the items in the warehouse and give instructions about their items, such as where and when they want the items.

In this case study, one process of a warehouse is presented. The process, 'manual redistribution within a warehouse and between different warehouses', is proposed based on a use case of the warehouse case study in Jacobson's OOSE book. Appendix B illustrates the possible details of the manual distribution process prior to the introduction of a computer system. That is, before automatising parts of

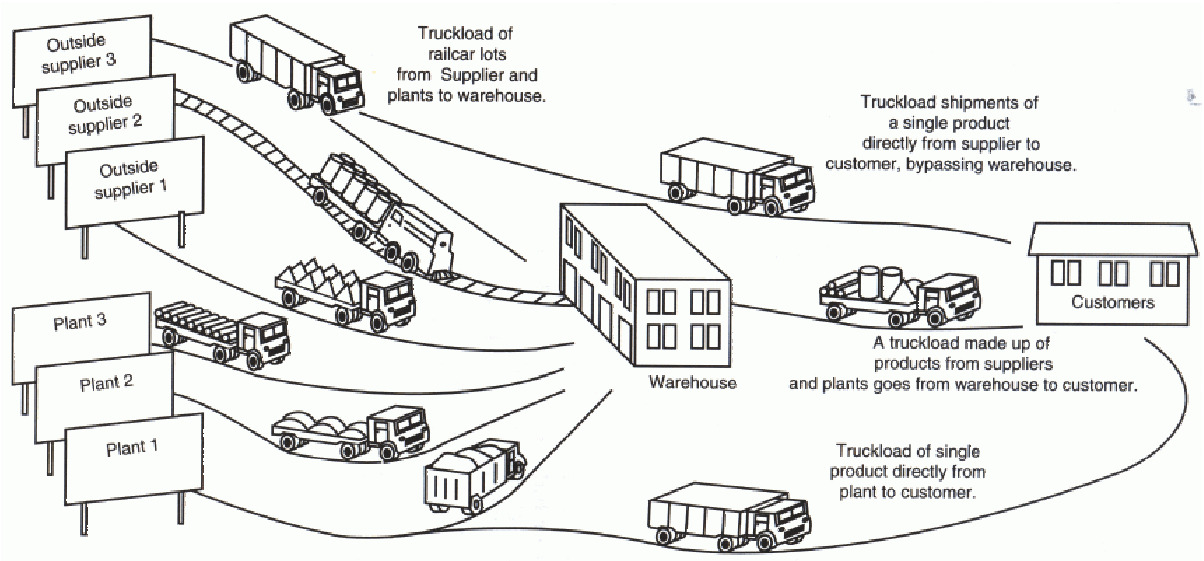

Figure 7.4 The Overview of a Warehouse (adapted from Johnson and Wood, 1996)

5. Such as industrial chemicals and foods.
6. The possibility that some of the functions of the warehouse associated with the physical storage and retrieval of items might also be automated using robots is not beyond the scope of our approach, but this is not directly addressed here.

the process by computer systems, the information needed for manual distribution tasks would be done through filling in and distributing different kinds of forms as well as the use of public inventory boards. Other methods of communication may be used such as telephones, but the delivery and completion of paper-based forms would still be the main method for the sake of authorisation and auditing. Another reason we chose form delivery to abstractly represent the process is that the forms, and the relevant information recorded on the forms, are obvious observables to identify the current situation of the warehouse. Figure 7.6 shows a snapshot of the process appearing on the window of the ISM.

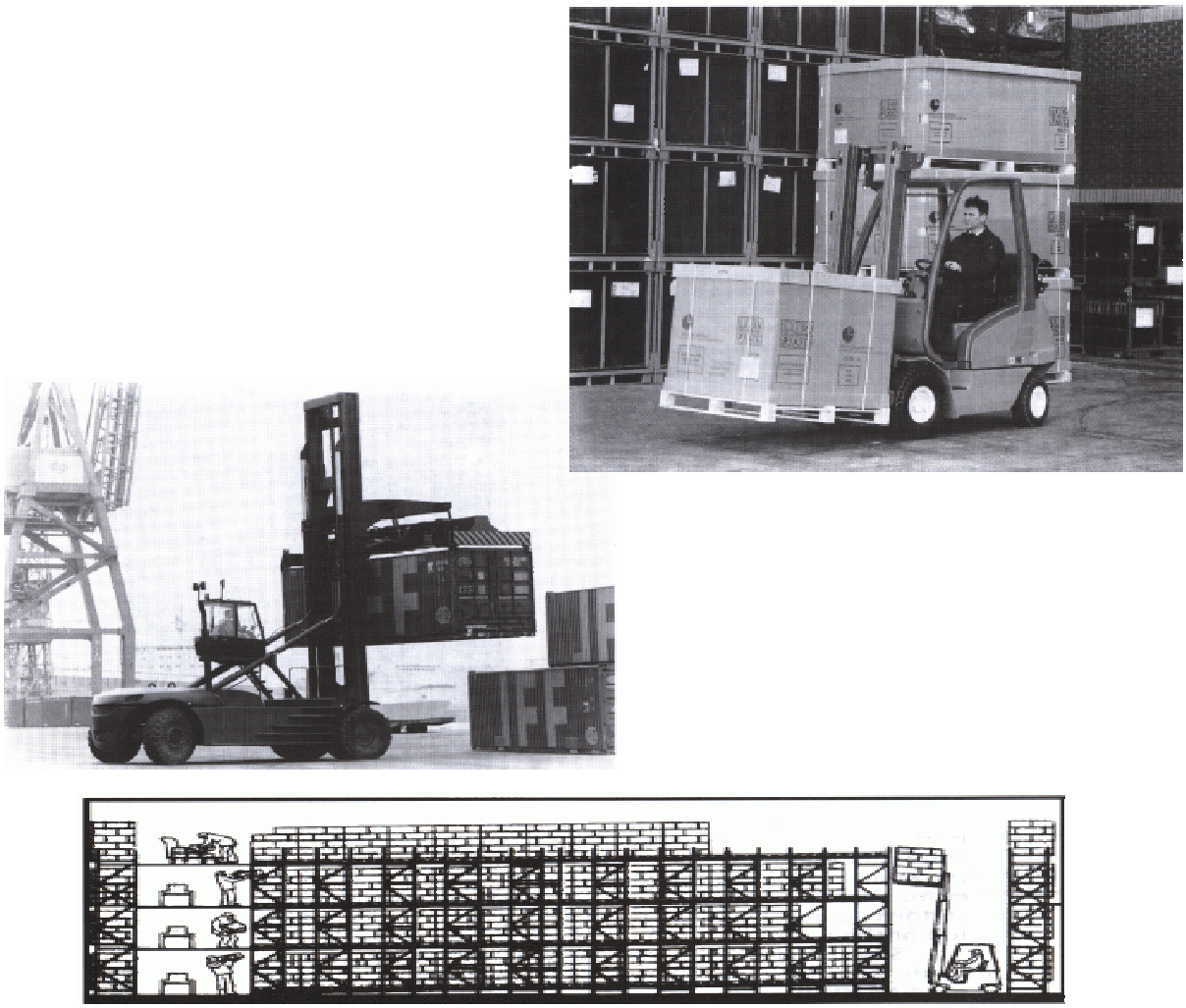

Figure 7.5 Forklift Truck (from Rushton et al., 1989; Johnson and Wood, 1996)

7.3.2 Business Process Model for Warehouse

In applying the use case approach, the first step is to create a simple picture of a *system* which describes the system boundaries and the actors (i.e. the users) of the system (cf. Figure 7.7). The significant difference between the EM approach and the use case approach is that the boundary of the system is not preconceived but grows with the increasing understanding of the modeller. In conventional system development, because the boundary is defined in advance, the modeller focuses on those interactions which respect the functionality that will be imposed in the proposed system. In the EM approach, the warehouse operation is conceived in terms of each agent's perception of states and state changes. For this reason, our initial concern in developing the business process model is with studying the capabilities of the agents that are identified, and examining the possibilities for their unconstrained interaction in an experimental manner. This will form the basis for subsequently exploring the protocols that these agents can follow in order to carry out the characteristic transactions of the warehouse.

In traditional modelling approaches for business, the issues of how agents relate to the current state of a business process, and the dependencies between entities, is not explicitly addressed. In object-ori-

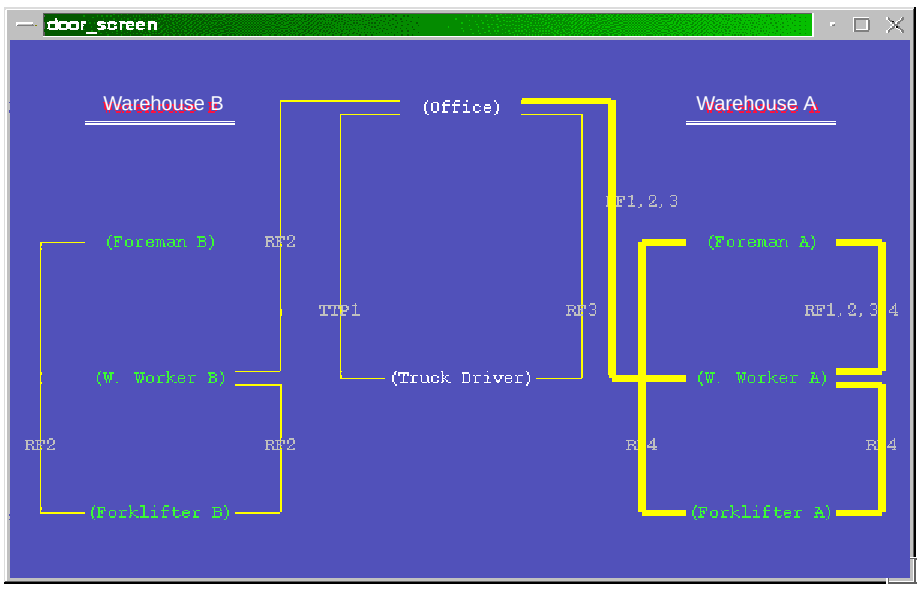

Figure 7.6 Snapshot of the Warehouse Process

ented approaches, for example, such relationships and dependencies are implicitly expressed through messages passed from one object to other objects which trigger the calling of methods with the objects to update their state. It may be easy to implement such kinds of dependency within the object-oriented paradigm, but it requires all the dependencies represented in the model to be preconceived prior to implementation and the message passing used to control the update requires specified ordering. Further, if the messages are not sent at the correct time to trigger the update, then the state of the model will become inconsistent.

The development of process models from our EM approach has two aspects: an observation-oriented analysis and an associated simulation of behaviour. The latter is based on the participation of human agents interacting with the computer model, and represents a shift in emphasis from exploratory activity and model extension to the study of the intended process operation (Chen et al., 2000b). The key issues are the identification of stable protocols for interaction and suitable stimulus-response patterns.

To model a business, there exists some key observables that are recognised to be strongly related to both the daily work of personnel within the business and the phases in the preconceived transactions in a process. For example in this warehouse case study, these include the items stored in the ware-

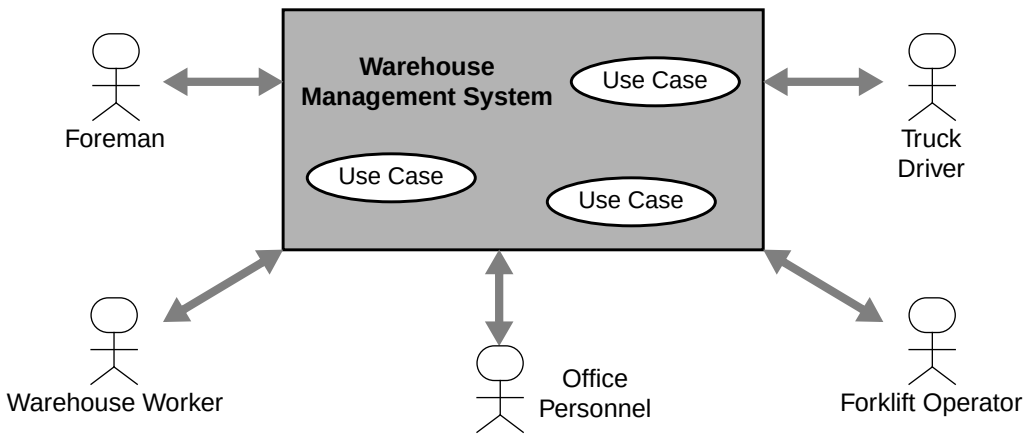

Figure 7.7 Diagram of the Initial Warehouse System with Actors Identified (adapted from Jacobson et al., 1992)

house and their locations. Our EM approach pays more attention to the true character of the real-world observation. As a result, the role of observations in the business process differs from a conventional approach in three important respects (Chen et al., 2000a):

- Firstly, the way in which states and events are observed by an agent is considered to be significantly important. It is not simply the fact that an item is at a location or that a truck has arrived at the warehouse that is regarded significant; it also matters how and by whom the presence of an item or the arrival of a truck is, or can, be observed.
- Secondly, the fact that agents are aware of the abstract stage that has been reached in the business process is taken into account in identifying their observables. For example, the office personnel will distinguish the abstract status of a redistribution process according to whether a group of items is still at the warehouse, in transit or has been successfully relocated.
- Thirdly, there has to be a means by which agents can interpret physical observation of the real-world state as disclosing the status of abstract business transactions. For example, there must be some concrete indication which points out that an item has now officially left the warehouse and that a new phase in the redistribution process has begun.

Our EM business model is built with reference to state change of an abstract nature that is associated with observation of a process. In the warehouse example, the relevant observables are related to the current status of the communicating information about the warehouse operations between agents. The corresponding state changes are concerned with the systematic execution of protocols and the associated transition from one phase to the next phase. By using the state changes as a representation for business processes, we can easily identify the existing processes and – from potential problems or dissatisfactions of customers or employees – can also find those that are candidates for the BPR activity. An important aspect of the observables in our business model is that they should not only serve to determine the current state, but also supply a transaction history appropriate for auditing (Chen et al., 2000a).

In order to understand the existing business processes properly, the ISM we develop to represent the business process model is modelled on the practices that would have been used in the operation of the warehouse prior to the introduction of computer systems. This is consistent with the emphasis of Jacobson et al. (1995) on the benefits of modelling existing practice[7]. In this context, forms and paper delivery serve as records of the operation of the model. This kind of manual data entry following systematic processes of form delivery can represent both the current status of all transactions such as which items were in transit, and the history of transactions. The aim of BPR is to automate these transactions by introducing computer systems, and to try to find alternative ones which will reduce the work-hours of personnel and achieve a more efficient process for business.

From our viewpoint, the forms can be interpreted as a *paper-based* ISM for the business process. In carrying out a particular transaction, specified procedures are followed in filling forms and transferring them between warehouse agents. For example, as depicted in Figure 7.8, when a manual redistribution between warehouse is initiated, four copies of redistribution forms (RFs) are transferred from the foreman to the warehouse worker. The manual activities of processing forms can effectively identify which agents have the roles in the transaction, which are currently active in any phase, and how their interaction is synchronised (cf. Figures 7.8 and 7.9). The current status of any transaction is determined by what sections of the forms are currently completed and who currently holds the forms.

7. They emphasise the significance of analysing and modelling the existing business in a chapter (“Reversing the existing business”) in their book (Jacobson et al., 1995).

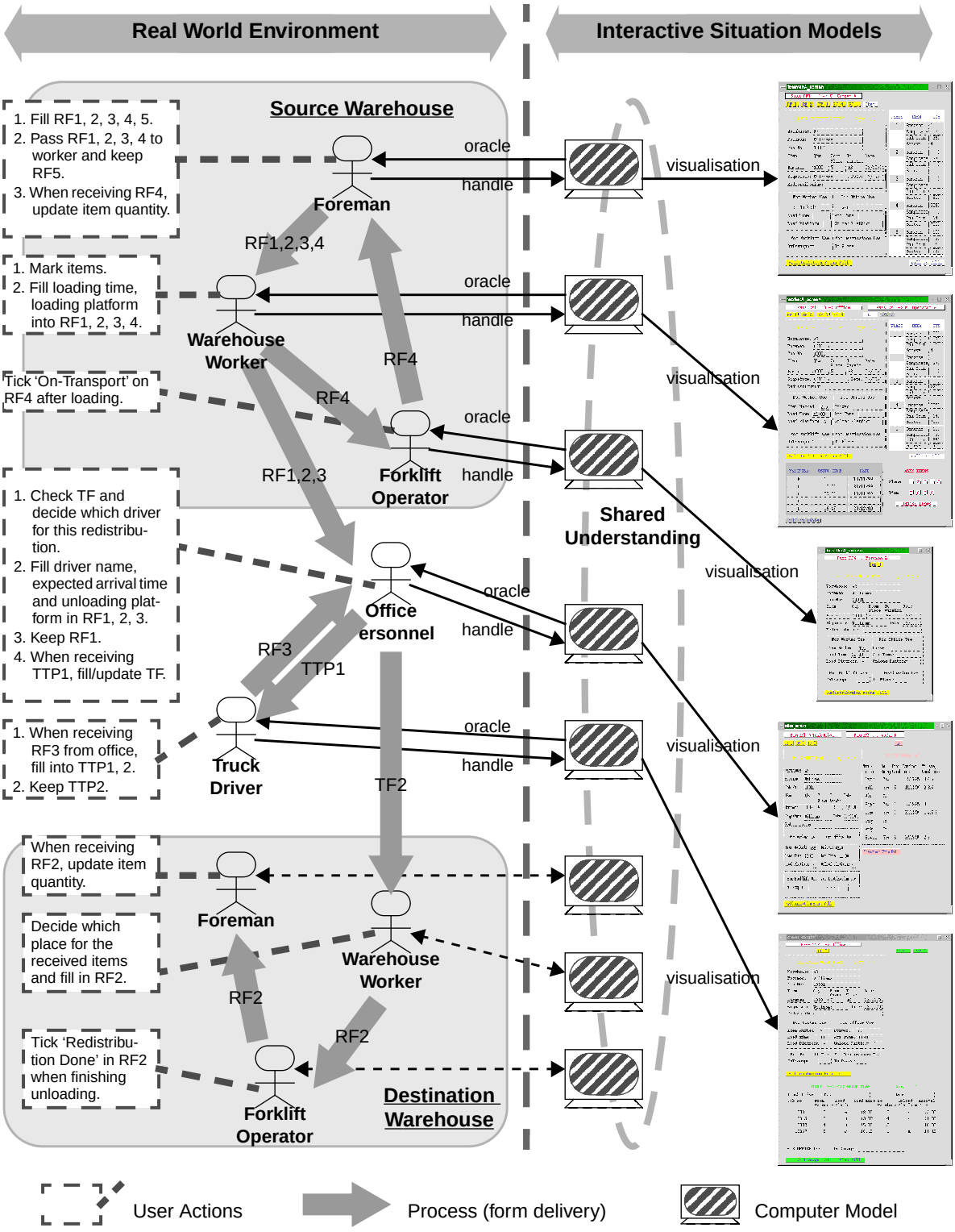

Figure 7.8 A Collaborative Working Environment for Manual Redistribution Between Warehouses

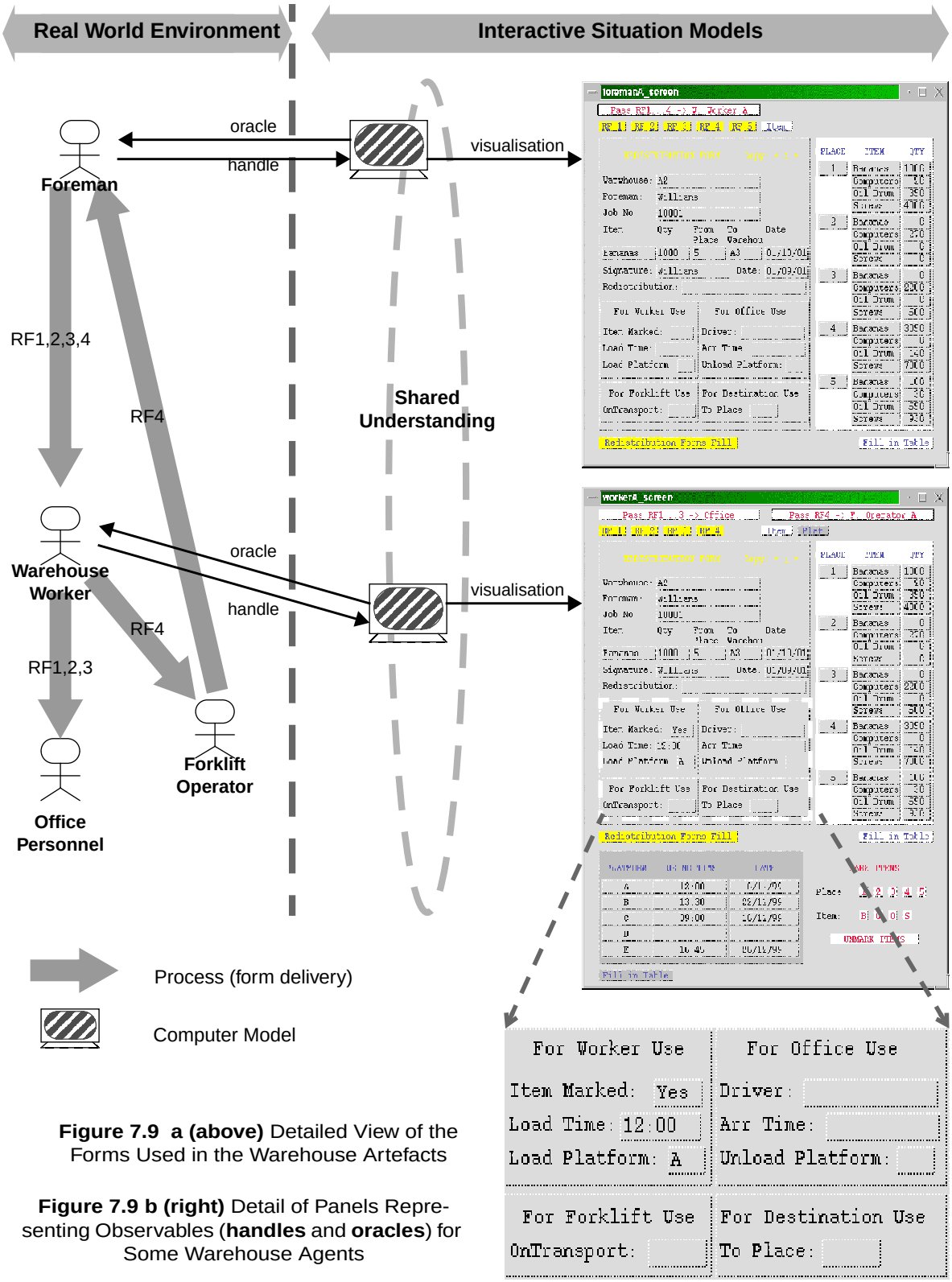

Figure 7.9 a (above) Detailed View of the Forms Used in the Warehouse Artefacts

Figure 7.9 b (right) Detail of Panels Representing Observables (**handles** and **oracles**) for Some Warehouse Agents

The modifications which the warehouse agents make to the forms and the movement of the forms themselves is abstractly represented by the path traced through the business process. In order to explain this in detail, we need to refer to the observational and interactional context for agents, that is, the observables they can refer to (the *oracles*), those they can conditionally change (the *handles*), and the protocol that connects both the oracles and handles (cf. subsection 4.2.2). Further, the relevant observables in this context can refer to the state of the warehouse (for example, an item can be signed off only if it is presently to hand) and relate to a high-level context for interpretation (for example, issues of legality or safety). This information plays an essential role for the purpose of auditing and traceability.

Appendix A shows the LSD account of this observational and interactional context of the warehouse. In this account, the interpretation of agent actions may vary due to the current status of the business process which is being investigated and the modeller's current (and subjective) understanding of that situation. For example, Beynon (1997) and Ness (1997) identify three views of agents for different contexts at different phases. In the early stage of familiarisation with an environment or 'putative' system, an agent has unexplored potential to affect system[8] state (view 1). At the later stage, an agent may be construed as reliably following some particular patterns of stimuli-response within the system (view 2). When an appropriate business process has been successfully identified and implemented, each agent enacts a pattern of stimuli-response interaction which can be entirely circumscribed and predicted (view 3). In view 1 our concern is whether an entity has any influence over its environment and in view 3 our concern is whether the exact nature of the influence is known. The term 'agent-oriented' is commonly used to refer to activities that are interpreted from the view 3 perspective, but in EM the concept of agency can only be meaningful when relating to the process of moving from view 1 to view 3 perspectives.

In the warehouse case study, there are contexts in which all the human agents can be viewed in one of these three perspectives. When the modeller is initially unfamiliar with the environment and the

8. The 'system' in our warehouse case study is regarded as the whole organisation (the business system), not only the computer system itself.

processes of the business, the personnel will represent the example of view 1 agents whose interaction with the warehouse environment and the operation of the business is unexplored. When the roles of a particular employee, such as the foreman, have been more clearly identified, they can be regarded as the view 2 agents as whose pattern of stimuli-response can be in some levels clearly identified. The aim of our modelling process is to fully understand the whole business process and attribute automated agency to the view 3 agents whose pattern of stimuli-response is entirely predictable. As described in Chen et al. (2000a), this accords with our argument that the business model is a form of generalised program, and the activity of business process reengineering closely resembles the capture of the program requirements.

The LSD account can be viewed as identifying and classifying the observables that reflect the modeller's current understanding and subjective viewpoint on the warehouse operation. When the construction of the model through the modeller's experimental interaction follows the transition from the view 1 to view 3 perspective, the LSD account can then be regarded as specifying the observables that describe the stimulus-response patterns in the organisation. We have described in chapter 4 the various kinds of observables in an LSD account. The observables which are attached to an agent are referred to as *state* observable. In general, the values of these observables can be directly changed by other agents. For example, the agent **`warehouseWorker`** can change the status of an item to 'moving pending' by manipulating the **`rf_moving_pending`** observable which is a state for the agent **`rf`**. The *oracles* are the observables to which an agent responds. For example, **`rf_item`** and **`rf_quantity`** in the agent **`warehouseWorker`** are examples of oracles for the warehouse worker, who has to know the identity and quantity of items to be redistributed before changing their status (cf. Figure 7.9 and Appendix A). The *handles* for an agent are those observables that are conditionally under its control. The observable **`rf_moving_pending`** is an example of a handle for the agent **`warehouseWorker`**.

The stimulus-response patterns for an LSD agent are modelled in two ways. The *derivates* are used to represent stimulus-response relationships that are indivisibly coupled. For example, the observable **`transportationError`**, which indicates whether there is a truck available for the specific time at which the foreman intends to make redistribution, is a *state* for the agent **`environment`** but also a *deri-*

vate for the agent **`foreman`**. That is to say, any change in the status of truck availability will also simultaneously change this observable. Looser coupling of stimulus and response is modelled in *protocols*, which consist of a set of guarded actions, each of which takes the form of an enabling condition and an associated sequence of possible redefinitions of observables. Each 'guarded action' can be regarded as a privilege to act. That is, if an enabling condition is met, a particular action *may* be performed. As an example of this principle, the agent **`warehouseWorker`** receives redistribution forms from the foreman (the enabling condition), then decides the loading time and platform, and passes the forms to both the office personnel and forklift operator (the guarded action).

Appendices B and C give the details of the warehouse process and Figure 7.10 shows a diagrammatic summary derived from the LSD account of the interactions between all the agents participating in the warehouse processes. The tables (which represent the information shown in the paper-based forms or on the boards) detailed in Appendix C represent all the artefacts which are necessary for the completion of redistribution among warehouses for a specific day. And Appendix B gives the details about the actions of different agents in terms of oracles and handles (i.e. the observables in the tables) as well as the dependencies between these observables. The diagram depicted in Figure 7.10 represents an abstract level of the warehouse process by means of form (tables) delivery[9] and the access to, or observation of, forms or boards by different agents. For example in Figure 7.10, the oracles to the warehouse worker are represented in the associated family of tables, where Tables F and G feature in Figure 7.12 below.

7.3.3 The ISMs for Representing the State in the Warehouse

One of the consequences of using EM principles is that the construction of computer-based artefacts has an explanatory role in that construction process. The state of each artefact is specified by a definitive script and is intended to represent the current state of its referent as viewed by a particular internal or external agent (cf. Figure 4.2). For the warehouse example, the relevant internal agents include fore-

9. The concrete details of form delivery have been depicted in Figures 7.8. and 7.9.

men, warehouse workers, forklift operators, office personal and truck drivers. The possible external agent can be an 'omniscient' global observer who is able to see all the movement and interaction within the warehouse (like the role of a God-view), or an auditor whose task is to trace transactions. By networking these artefacts and establishing dependencies and interactions among them we can represent the state of the relevant domain which reflects the related views of agents.

For interpreting the business process model and ensuring that the abstract phases of the model can be appropriately embodied in an agent's perception and action, we need to take account of the explicit and physical observables associated with the operation of the warehouse. In order to construct the business model under the SPORE framework in a situated manner, the ISM is used to incorporate

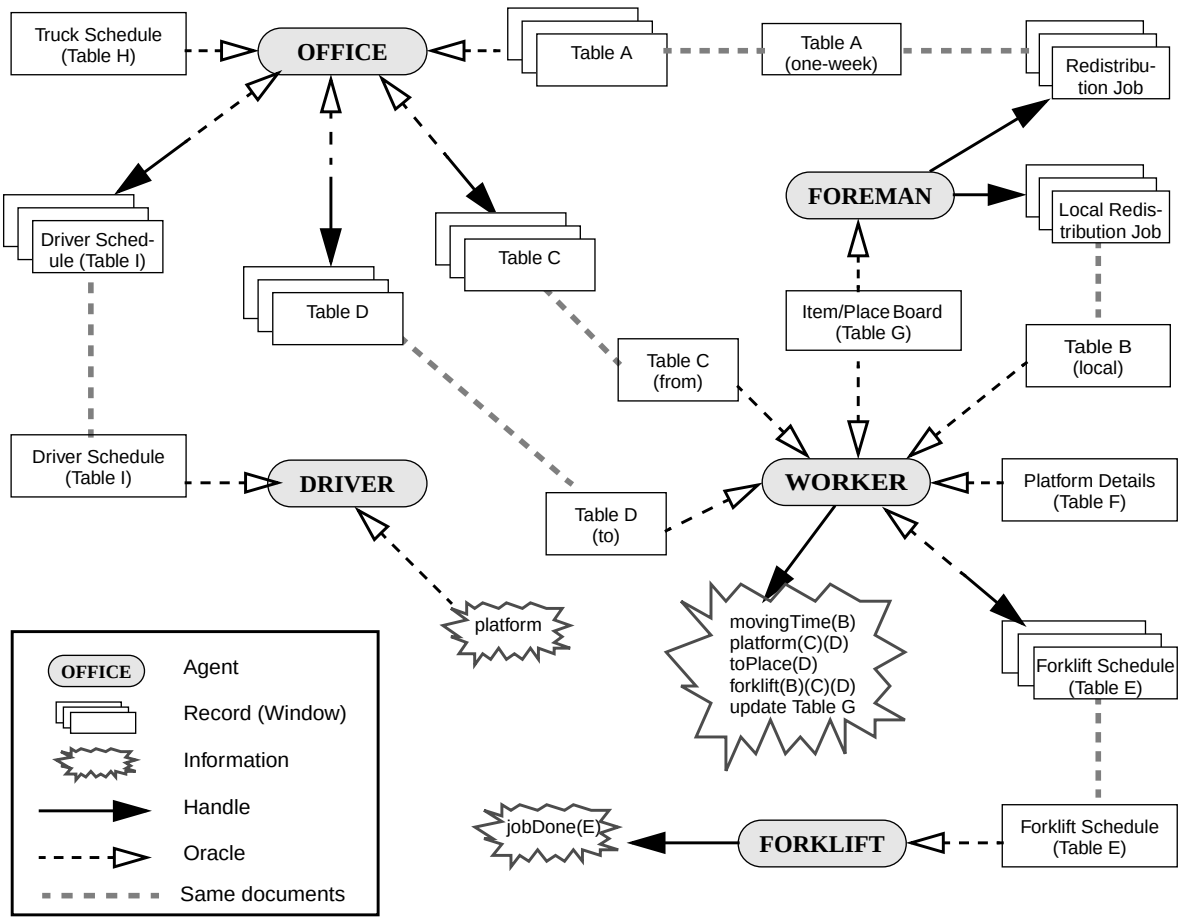

Figure 7.10 A Diagrammatic Summary of an LSD Account of Warehouse Processes

the ‘matter-of-fact’ observations of the current state of the warehouse. As mentioned earlier, the typical observables which may be significant are the items and the locations in the warehouse, as well as the inventory which records (connects) both the items and locations. The ISM we developed which represents these observables can supply the visual representations for items and locations, and also display the status of the inventory. Figures 7.11 and 7.12 are screenshots from artefacts created to represent several aspects of state in the warehouse situation, as seen by a warehouse worker. The left-hand side of Figure 7.11 depicts the physical state of the warehouse, showing the layout of the storage places and transportation platforms, and the location of items and information boards. The right-hand side depicts a form (the redistribution form) relating to the status of individual items in a redistribution process. Figure 7.12 depicts two of the many tables of information that together give the warehouse worker a more comprehensive view of the current status of items in the warehouse. All these tables have been described in Appendix C.

Such a representation of the current warehouse state is complemented by informal actions, for example the relocation of items, item looking-up in the inventory or the reception of a new item for stor-

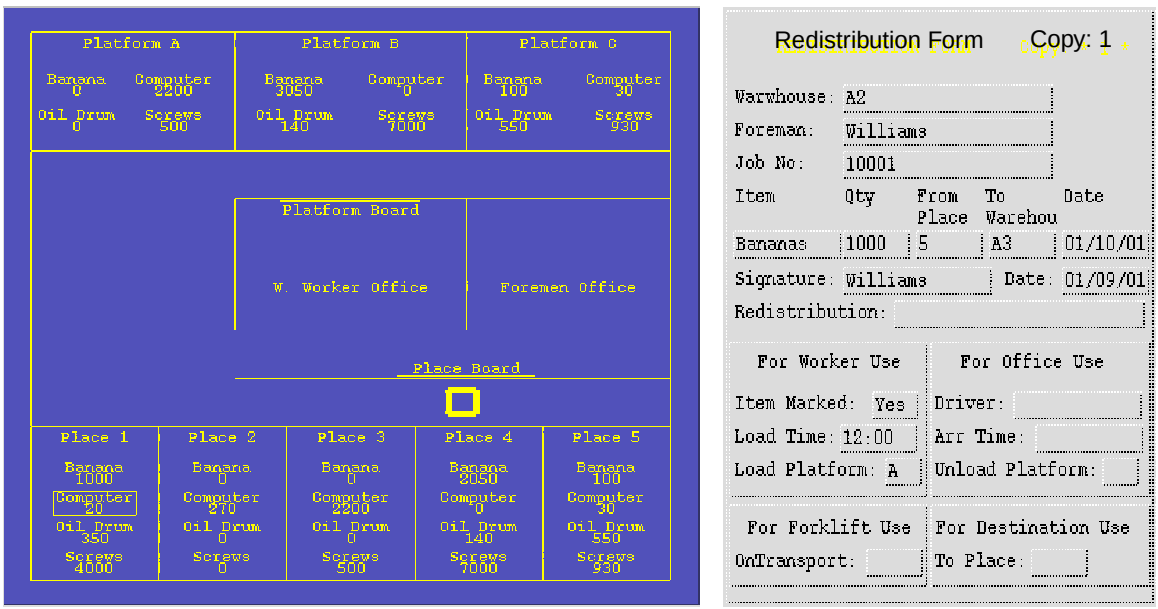

Figure 7.11 (left) Physical State of Warehouse and **(right)** Form Showing Individual Item Status

age. And these different aspects of state play their part in shaping the role of particular workers in the warehouse. For example, the warehouse worker will be aware of all these aspects when making decisions, bearing in mind the physical locations when dealing with items (Figure 7.11 left); knowing the current status of the transaction through examining forms (Figure 7.11 right); and consulting the work schedules and timetables when planning the movement of items (Figure 7.12). In some contexts, this will motivate visualisations representing the intermediate states in the warehouse operation, for example the items in transit, or items located via the inventory but having not been retrieved from the warehouse.

Of course there is much more to the state of the warehouse than those aspects of state mentioned above can express. For example, checking the status of items will typically involve communication with other personnel. Also the architecture of the warehouse, the locations of information boards and the organisation of items all have a significant impact on the activities of the warehouse process. There are

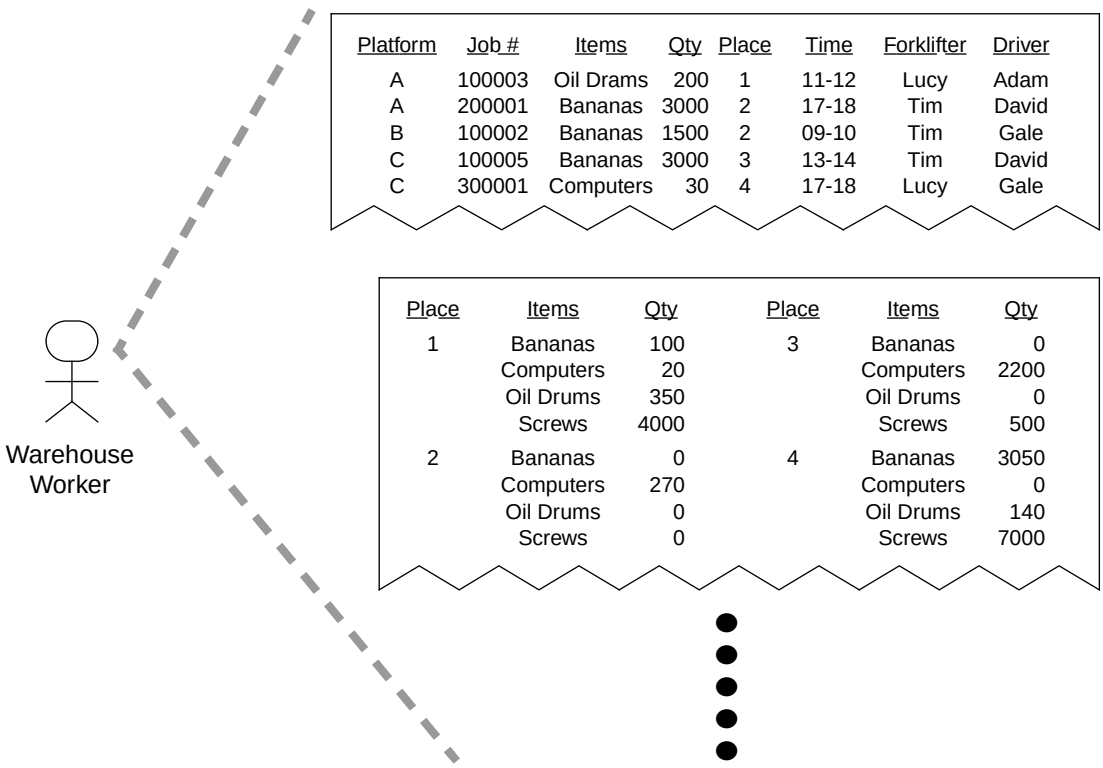

Figure 7.12 Tables Showing Comprehensive Item Status

also other issues about the way information is stored and transmitted on the paper-based forms. For example, there are security issues: generally the paper forms are physically inaccessible and are hard to modify without detection and can be audited. There are many additional significant aspects of states which are not represented above, for example the geographic location of trucks, drivers, other warehouses, and other factors such as traffic conditions.

From our viewpoint, a model of the warehouse should incorporate as much as possible of such aspects of state and state changes in order to correspond faithfully to its referent. It should also provide a context to experience the behaviours which are unpredictable and outside the scope of normal operation. Relevant observables for this purpose in the warehouse case study may include the issues such as the loss of items or warehouse locations, the concept of items being mislaid, or of items being perishable. The significance of our approach to the construction of models is that there is flexibility to adapt state interactively through cooperative activity in a distributed environment. This is achieved through the EM paradigm for state representation, which not only allows the state of the artefact to be changed by redefining variables[10], but is also open to allow the participants to *reinterpret* this state both through a shift in an individual participant's perception and through negotiation among these participants.

The interpretation of the LSD account (Appendix A) is closely tied up with the experimental interaction and investigation of the artefacts which are depicted in Figures 7.11 and 7.12. In the manual warehouse system, the dependency between contexts of tables represented in the LSD account might be maintained by directly copying from one table to another, or by the use of carbon copying. In an automated environment with supporting information systems, this dependency can be achieved by the use of relational databases with regular updates. We have suggested in subsection 6.4.1 that the organisational relationships in a business can be understood from the LSD account. That is, the LSD account can be viewed as a conceptualisation of agency and an expression of the agent's abstract perception of state. Provided that this is consistent with the physical and perceptual situations, such a LSD account

10.The redefinition of variables in this context can be interpreted as a state-transition, a correction, or extension of the artefact.

can be used as the basis for system redesign and implementation. For example, how the oracles of one agent are linked to the handles of another agent in Figure 7.10 indicates how this LSD account supplies a framework for defining the warehouse processes. The protocol, oracles and handles in the LSD account of an agent such as the warehouse worker indicates how the interfaces required to carry out the actions of its protocol are defined.

7.3.4 The Development of Warehouse Management System

This subsection will introduce the final system for warehouse management developed by this author and describe the process of its development which aims to link this practical work with the theoretical framework developed in chapter 6. As a definitive database management system (the EDDI) has been used in the development of the final system, this subsection will give a brief introduction to this database interpreter before describing the warehouse management system.

Introduction to EDDI

EDDI (Eden Definition Database Interpreter) is a definitive database notation which is implemented by the EM tool as a database management system. The notation of EDDI was developed to illustrate how the definitive scripts can be used to set up and query a relational database with the characteristics of the models by Codd (1970; 1979) and Todd (1976)[11]. Within EDDI, the database and the concept of dependency can be combined and this demonstrates a powerful character by applying the EM principles to a database structure. As implemented by the EM tools, the tables defined under EDDI are themselves made with definitions and thus they will automatically update when the contents of tables change. A database table under EDDI can be defined through a definition which maintains the dependencies in the script. This means that any change in the script will automatically update ('propagate' to) all the relevant observables which directly or indirectly dependent on it. This situation can be illustrated

11. Codd's relational model allows attributes to be organised in a systematic way according to the relationship between them. Tood implements a system to maintain relationships between data (based on Codd's relational model) by using dependency.

through a running script depicted in Figure 7.13. The tables shown in Figure 7.13 are created with the EDDI commands (in the EDEN input window), for example:

```
%eddi
Coventry (JobNo int, FromHouse char, Foreman char, Items char, Qty int, FromPlace int, ToWarehouse
      char, FinishedBy char);
Coventry << [100001,"Coventry","Bill","Screws",550, 4,"London","30/11/01"];
Coventry << [100002,"Coventry","Susan","Bananas",1500,2,"London","10/11/01"];
Coventry << [100003,"Coventry","Bill","Oil Drums",200,1,"Manchester","20/11/01"];
Coventry << [100004,"Coventry","Susan","Computers",120,5,"London","20/11/01"];
```

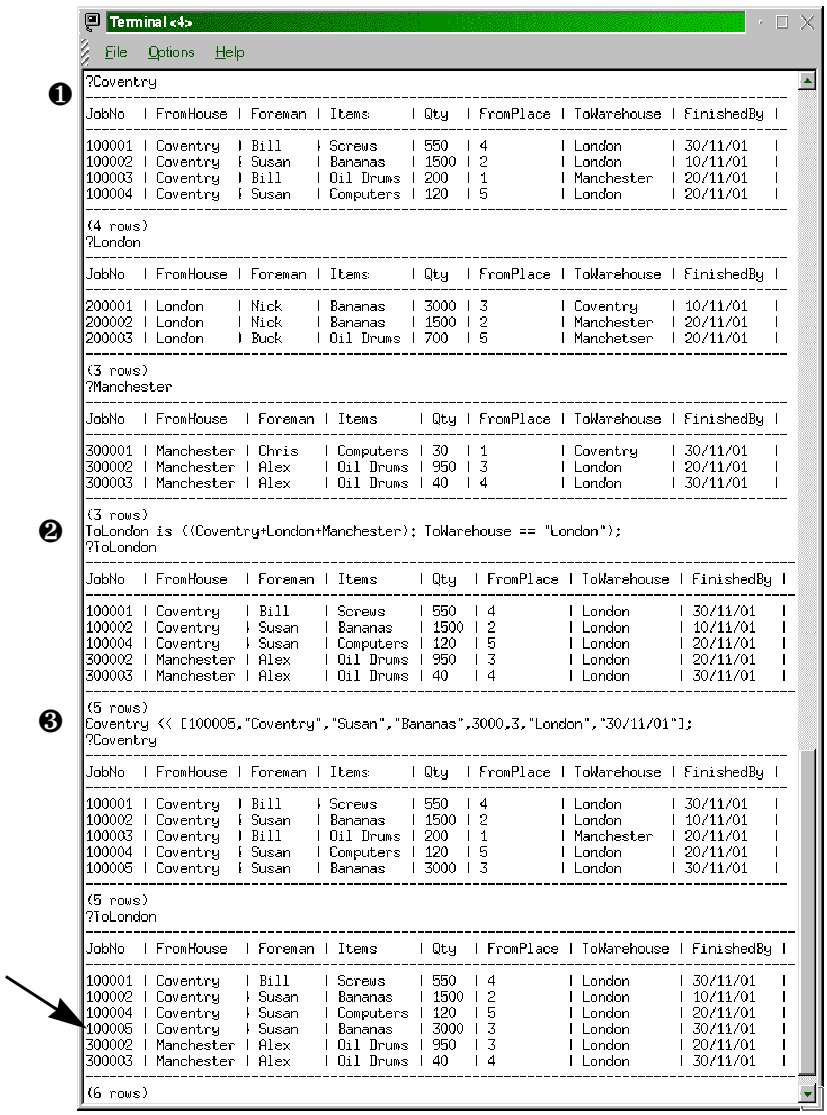

Figure 7.13 A Snapshot of EDDI Script

will create the table `Coventry` and insert four tuples with typed fields into it (marked ❶ in Figure 7.13). The *insert* command '**<<**' followed by a tuple separated by commas enables the modeller to insert tuples into a table at any stage (cf. mark ❸). Tuples can also be deleted from a table by *delete* command '**!!**' such as

```
Coventry !! [100004,"Coventry","Susan","Computers",120,5,"London","20/11/01"];
```

The query command '**?**' followed by a specific table name enables the modeller to view that specific table, especially to inspect the result of a relational algebra expression or the effect of any change in the script due to the dependencies. For example, running '?`Coventry`' in the EDEN input window will display the contents of the table Coventry (cf. mark ❶).

The EDDI interpreter also provides other commands for data manipulation and data definition. There are several operators of relational algebra which, as identified by Codd (1979), are used to specify the definitions for existing tables or refer to views: *union* (**+**), *difference* (**-**), *intersection* (**.**), *projection* (**%**), *selection* (**:**) and *join* (*****)[12]. The definition can be defined at any time in the script. For example in Figure 7.13, the specification of a definition (the line marked ❷)

```
ToLondon is ((Coventry+London+Manchester): ToWarehouse == "London");
```

will define a definition and create a new table `ToLondon` which will consist of the tuples found in either table `Coventry`, `London` or `Manchester` (under the *union* operand '**+**') and whose destination warehouse is London (by the *selection* operand '**:**' to compare the tuples which satisfy the specific condition). The word '**is**', as described in subsection 4.2.1, indicates that the dependency is maintained automatically. So the value of left-hand side (`ToLondon`) is maintained to be the up-to-date value of the relational expression on the right-hand side. We can illustrate the effect of the definition within EDDI by the same script in Figure 7.13. Before adding the new tuple into the table `Coventry` (i.e. mark ❸), the table `ToLondon` consists of all the relevant tuples which are to be found in the three existing tables.

12.These operators are in increasing order of precedence. It is only possible to construct the *union*, *difference* and *intersection* of two relation tables if they have the same type.

After inserting a new tuple into the table `Coventry` (i.e. the job number: 100005), not only this new inserted tuple appears in the table `Coventry`, it also appears in the table `ToLondon` (the dependant). This is because it depends on the table `Coventry`, and also the new inserted tuple has satisfied the condition set in that definition.

Introduction to the Useful System

In this warehouse case study we assume a simple situation in which the warehouse management company has a main office (as its headquarters) and three warehouses located in Coventry, London and Manchester. The following people will be using the system: the office personnel and truck drivers, and in each warehouse the foremen, warehouse workers and forklift operators. The system used is formed by several computer-based models (the ISMs) running in a distributed environment. There are three main artefacts in each model presented to the human agent: the user interface window for giving a command and inputting/modifying data; the terminal displaying the relevant information, in the format of EDDI database, to respond to the human action; and the EDEN input window which enables the human agent to send messages to other models and, provided such that human agent has an appropriate privilege, to add or modify the definitions in the script.

In Figure 7.14 the upper part (A and B) shows the window management system for the foreman in the Coventry warehouse, the lower part (C, D and E) shows the window management system for the office personnel. Figure 7.14 (A) is the window presented to the foreman in which he can give a command for redistribution between warehouses, either by adding new redistribution jobs or modifying existing ones[13]. Figure 7.14 (B) is the terminal windows which shows two tables in the EDDI format: one shows the current status of items in the warehouse[14] (when the foreman clicks on the button marked ❶ in (A)), and the other shows all the redistribution jobs which have been planned in the warehouse Coventry[15] (i.e. the button and table marked ❷).

13.The foreman can input the job number of an existing job in the window and modify its contents.
14.Compare with the Table G in Appendix C and in Figure 7.10.
15.Compare with the Table A-1 in Appendix C and Table A in Figure 7.10.

Figure 7.14 (C), (D), (E) show the windows for the office personnel as well as the relevant EDDI tables on the terminal[16]. As the main work of office personnel is to plan a truck drivers' schedule and the date for a specific distribution job, there are several information formats which the office personnel may find useful and helpful for their decision making:

- The office personnel may like to view all the distribution jobs from any specific warehouse (for example they can view all the jobs made from the warehouse Coventry (mark ❶) or from Manchester (mark ❷)). Alternatively, they can choose to view all the distribution jobs whose destinations are the same warehouse (for example, all distribution jobs to the warehouse London (mark ❸)). The information represented here can help the office personnel to decide the most economical truck schedule (for example, allocating the same truck for two redistribution jobs from Manchester to London and from Coventry to London) and avoid the allocation of two trucks between the same two warehouses on the same day.
- The office personnel may sometimes need to view all the redistribution jobs which need to be finished by a specific date (mark ❹).
- When deciding a truck for a redistribution job, the office personnel also need to check the time schedule of all drivers (mark ❺). They can either view the individual schedule of a driver[17], or they can input the date in the window to view all the time schedules for drivers who have been allocated jobs on that specific date[18].

Note that the table showing all the redistribution jobs to London (i.e. mark ❸) is the same table of ToLondon described in Figure 7.13 (but using different definition with different attributes listed after *projection* '%'). In reality, all the tables represented here are defined under EDDI and use different definitions which have been described in the previous subsection. For this reason, inserting any new tuples or modifying the existing ones in one table will automatically update the contents of its dependent

16.Figure 7.14 (D) and (E) are representing the same information but running at different stages. We will use these two artefacts to illustrate the dependency feature in EM in this system.

17.Compare with Table I in Appendix C and in Figure 7.10.

18.Compare with Table H in Appendix C and in Figure 7.10.

tables. Using the same example illustrated in Figure 7.13, when the foreman in the warehouse Coventry gives a new command for redistribution (cf. Figure 7.14 (A)), this new job information (the tuple newly inserted into table `Coventry`) will appear in the relevant tables in the office model (cf. Figure 7.14 (E)) as such tables depend on the table `Coventry` and this newly inserted tuple satisfies the conditions set in their definitions (e.g. `FromWarehouse == "Coventry"` (Figure 7.14 (E) ❶) and `ToWarehouse == "London"` (Figure 7.14 (E) ❸)).

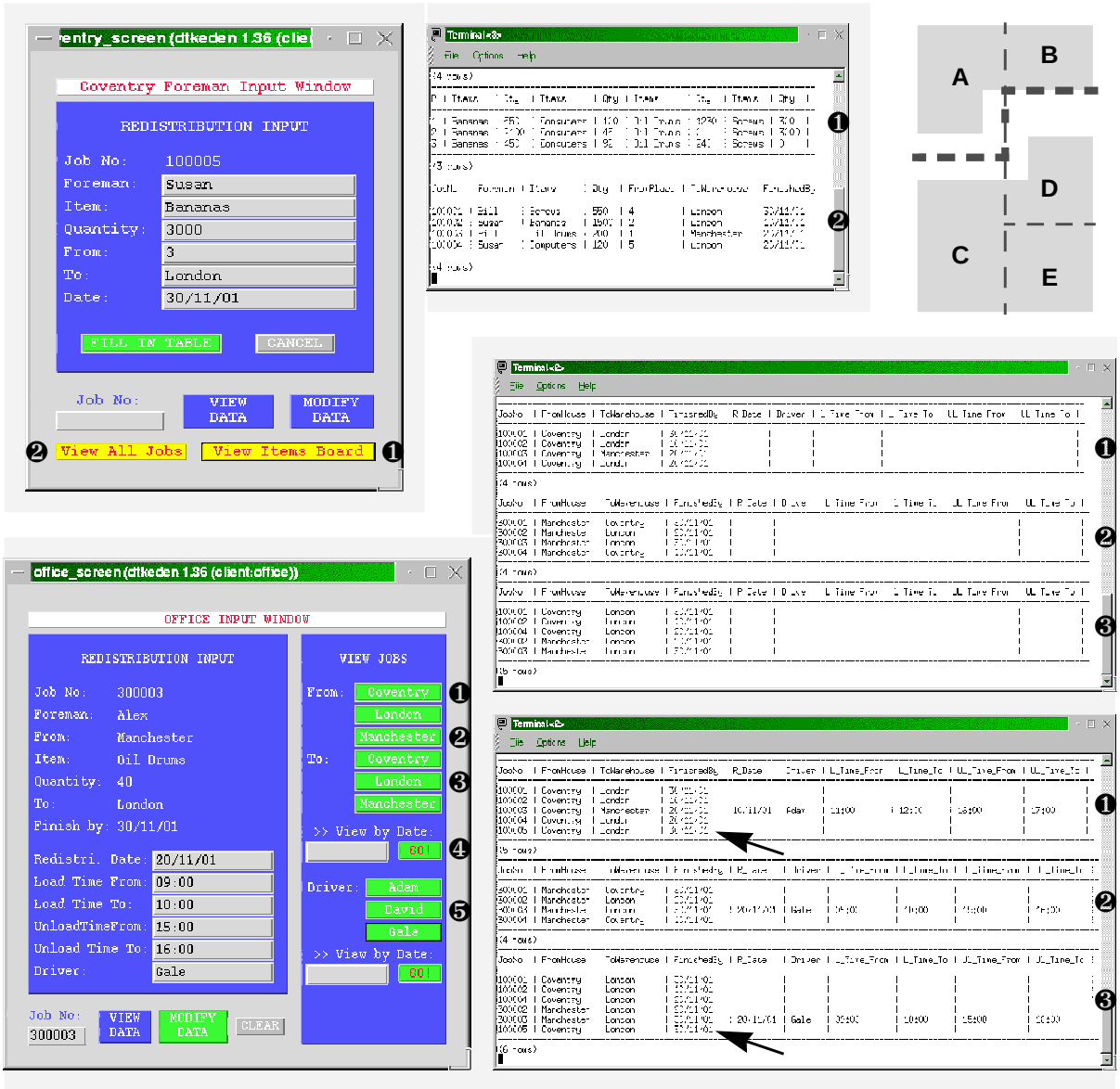

Figure 7.14 The Window Interfaces for Foreman and Office Personnel

Figure 7.15 depicts the window interface for the warehouse worker. Similar to the discussion above, there are also several information formats which the worker may find useful for their work, e.g. deciding a forklift operator for a redistribution jobs, allocating a loading or unloading platform, and finding out a suitable storage place for the items received. The worker may firstly need to know all the relevant jobs redistributed from, or sent to, his warehouse (mark ❶ in Figure 7.15). The worker needs to check the time schedule of each forklift operator in order to allocate the jobs[19] (mark ❷ and ❸). He also need to check the usage of platforms at a specific date[20] (mark ❺), and view the current status of item storage in that warehouse in order to decide a place for the items received (mark ❹). The worker can fill in the relevant data through his input window or modify the existing information at any time.

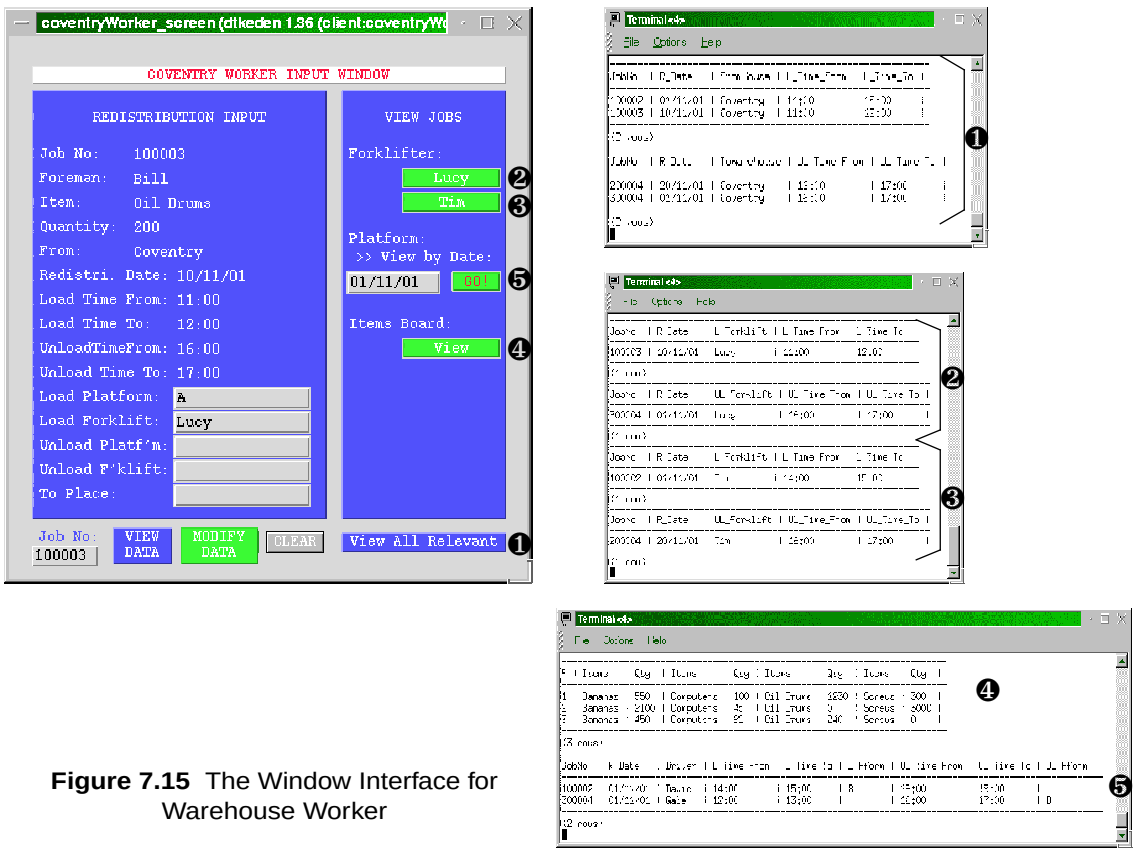

Figure 7.15 The Window Interface for Warehouse Worker

19.Compare with Table E in Appendix C and in Figure 7.10.
20.Compare with Table F in Appendix C and in Figure 7.10.

All the tables represented in the worker's window interface are also automatically updated whenever the contents of tables (the dependees) in office or foreman window change. For example, when the office personnel have input the driver and redistribution time for a job whose source or destination warehouse is Coventry, this job information will immediately be updated in the table within the Coventry worker's window due to the dependencies maintained by EDDI interpreter.

The Development Process of the Useful System

As described in subsection 6.4.2, there are two levels of contextual models in EM which enable the modeller to have contextual knowledge about the system to be developed or reengineered: the LSD account and the EM model. The LSD account in Appendix A records this author's perception about the observational and interactional context of a warehouse management system in terms of agency and observation. We have described earlier the various kinds of observables in that account. The identification and classification of these observables play a significant role in the author's development of the useful warehouse system, especially in the design of the window interfaces for warehouse people with different roles and responsibilities. For example, for an office person whose main work is to decide the redistribution date and allocate a truck for a job when receiving commands from warehouses. What he needs to know in order to make a decision for these may include: from which warehouses these jobs were proposed as well as their destinations, the date by which these jobs have to be done, the details of the time schedule for each truck, etc. Such information directly relates to his observations and has been recorded in the LSD account as *oracle* observables. The author's development of the window interface for the office personnel was guided by the identification and classification of the observables in LSD. For example, the *oracle* observables of agent `office` were directly mapped into the relevant information shown in the office window as well as the tables which provide the information for his decision making. The agent `office` has no privileges to modify such data, for example the kind of item and its quantity which was proposed by the foreman. This is because these observables were identified as *oracle* rather than *handle* in the LSD account. The *handle* observables, in contrast, gave the guidance when defining the input data within the office window such as the truck driver, the date and loading/unloading times for

a redistribution job. Also, as the *derivates* and *protocols* in LSD reflect the actual behaviour and action of human agents, referring to these helped this author to decide the appropriate buttons in the windows as well as define their functions.

Through the practical development of this system, the author found that the LSD account not only served as a contextual model for understanding the system environment, it also provided clues and guidance for choosing and defining the relevant data displayed in the interface and avoiding any omissions or redundancy. For example, the office personnel does not need to know in which place the item is currently stored, and a warehouse worker does not need to know which truck driver is allocated for a redistribution job or which warehouse is the destination when he decides the forklift and the platform for that job. Because these observables were not classified as *oracle* observables for such agents, through referring to the LSD account, they do not appear in the developed window interfaces. By analysis of this nature, possible omissions and redundancy can be avoided.

The second level of the contextual model is the EM model which embodies the contextual information described in LSD and enables the modeller to interact and have experience with it in an open-ended manner. The author has developed a business process model for the warehouse which has been described in subsection 7.3.2. This model (the paper-based ISM) can be regarded as the seed-ISMs in the unified development procedure in Figure 5.3. By interacting with this model, the author, as well as other participants playing various roles in the warehouse example, have gained a deeper insight and experience of the current redistribution process between warehouses, and understood how and why certain interactions were performed (cf. Figures 7.6 and 7.8). The visual displays of the paper-based ISMs are the mock-ups of the actual forms used in the warehouse (cf. Figures 7.9 and 7.11). Metaphorically 'filling in' and 'delivering' these forms in the ISM represents the current status of the process. The user windows of the final system were developed mostly based on the interface of these forms. The layout of the windows, as well as the definitions in the script, of the final system were modified mainly from the ones in the seed-ISM (through the advantage of the 'on-line' redefinition under the EM tools) to meet the actual need of relevant users. In this case, the ISM in the process of development plays a role as a rapid prototype which we described in section 6.4. The warehouse process prior to the introduction

of the system, i.e. the paper-based form delivery as depicted in Figure 7.6 and Figure 7.8, has been replaced (automated) by the dependencies maintained in the EDDI database. For example, the original delivery process of three redistribution forms (i.e. RF1,2,3 from the foreman to warehouse worker then to the office) has been replaced by the dependency between two tables: one in the foreman's interface (Figure 7.14 (B) ❷) and the other in the office interface (Figure 7.14 (D) and (E) ❶). Through the construction of, and interaction with, the ISMs the author can evaluate the system requirements to find out whether they were incomplete or inconsistent. For example, the system is required to immediately update whenever new data has been input. For this, it is necessary to check whether a new tuple has been inserted in the relevant table in the office interface when a foreman gives a new command through his input window.

The development process of the final system also reflects the essence of participative process modelling under SPORE. By interacting with the ISMs, the various roles of human agents (e.g. foremen, warehouse workers and office personnel) were participating and thus directly involved in the process of system development. The system was built under the distributed EM tool (the **dtkeden**) which formed a collaborative working environment as depicted in Figure 6.3 and Figure 7.8. Four interaction modes provided by **dtkeden** (cf. subsection 6.3.2) enable the participants to communicate with each other and 'visualise' their viewpoints in order to gain better consistency between their insights. For example under the *normal* mode, the foreman (probably with some specified privileges) might wish to express his idea of new requirements by modifying the query for an EDDI database in the warehouse window interface. The agent in the server model is able to see all the movement and interaction during the process (the God-view). This provides an opportunity for the auditor to be involved in the process or enables the senior management as well as the BPR designers to have a global view of the effect of any local change in a part of the system or the process. Decision making support and BPR for the warehouse management can also be achieved increasingly under this framework. As described in section 6.4, DSS and BPR are different mainly in their scopes of analysis, and DSS focuses on the perspective of internal agents whereas BPR focuses on that of external agents. In this final system, the office personnel or warehouse worker might wish to propose a protocol in their scripts in order to make their decision making more effi-

cient and accurate. For DSS, they can try their proposed protocol by adding or redefining the definitions in the script and test this in an interactive manner (the 'what-if' experiment). For example the warehouse worker can propose an procedure which will search for a forklift operator whose daily schedule is available for the redistribution jobs received and automatically fill the operator name in. From the perspective of BPR, the external agent in the server model can intervene in the process or give different privileges to the internal agents to view the global effect under the *interference* mode.

Participative BPR in the Warehouse Case Study

We emphasise in this thesis the importance of stakeholders' active participation in the process of BPR, and the distributed EM tool has provided a collaborative working environment for participative BPR. In this warehouse case study, besides the designers (including both the business designers and system designers) and the warehouse personnel mentioned earlier (foremen, officer personnel, warehouse workers, etc), the participants in the process may also include the senior managers and their customers. It is essential that senior managers are involved in the process because a critical factor to the success of BPR is support from top management; and customers must be involved as BPR is radically customer-oriented. And we include different roles of personnel in the process because they have different perceptions (and knowledge) about the business process they are currently using in as well as the newly designed ones. The following cases indicate some possible scenarios during the process of participative BPR, which include consideration of an individual viewpoint about the BPR plan proposed and the problems encountered.

Case 1 After using the management system, the foreman has experienced that it is easy to add and modify redistribution data through the interface, but it is difficult to identify the right place in which the items are currently stored. For example as Figure 7.14 (A) depicted, the foreman would like to move 3,000 boxes of bananas from Place 3 to Warehouse London. But actually there are not sufficient bananas in that place (i.e. only 450 boxes of bananas in Place 3 as depicted in Figure 7.14 (B)) for the command he proposed. During the process of BPR, the foreman can – perhaps with assistance by the

designer – add a new definition (in his ISM input window) to automatically display only the places and quantities of the item the foreman input in the interface. For example:

```
%eden
Proc showDetail_click : showDetail_mouse_1 {
   if (showDetail_mouse_1[2]==4) {
      itemName = itemText_getText();
      execute("%eddi\n ?PlaceDetailsCoventry: Items == \"" // str(itemName) //
         "\" %Items, P, Qty;");
   };
};
```

This will show an EDDI table in the foreman terminal which gives the information about the current storage places and quantities of the item the foreman inputs in his window interface. And the foreman can modify the existing process `addButton_click` (cf. Appendix D for the original script) to:

```
 proc addButton_click : addButton_mouse_1 {
    if (addButton_mouse_1[2]==4) {
       changeData();
       if (detectError()) {
          writeln("The input of the item’s quantity is not correct!");
          break;
       } else {
          execute(%eddi\n  Coventry << [\"Coventry\"," // str(jobNumberC)
             // ",\"" // c2 // "\",\""" // c3 // "\"," // c4 // "," // c5 // ",\"" // c6 // "\,\"""
             // c7 // "\,\"\"",\"\",\"\",\"\",\"\",\"\",\"\",\"\",\"\",\"\",\"\",\"\"];";
       jobNumberC++;
       };
    };
};
func detectError {
   for (i = 2; i <= PlaceDetailsCoventry#; i++) {
      if ((PlaceDetailsCoventry[i][2]==str(itemText_getText())) ||   /* Item Name */
         (PlaceDetailsCoventry[i][1]==(fromplaceText_getText()) ||   /* Place */
         (PlaceDetailsCoventry[i][3]<(quantityText_getText())) {     /* Quantity */
         return TRUE;
         break;
      } else {
         return FALSE;
      };
   };
};
```

Such a redefinition will enable the foreman to add or modify procedures/functions in the definitive script in an open-ended manner, i.e without rerunning or compiling the script. As these changes are localised within the foreman's ISM, other participants may all agree to this modification and thus include this in the new process.

Case 2 The office personnel find that the system makes their work easier and more effective than doing the same procedures by paper-form delivery. However they may find that the foremen in some warehouses send their redistribution commands to the office very late, and this always causes inconvenience for their daily work. During the process of BPR, they can experience a new situation by adding a new definition which will reject the foreman's command if the finish-by date is less than three days prior to today. Of course, such a change will meet resistance from the foreman when he sometimes receives the rejected commands from the office. With several 'communications' by redefining the definition, they may, for example, agree that the commands will be rejected if they are less than seven days prior to today.

Case 3 The truck drivers are sometimes annoyed by the situation that the redistributed items are not ready on the loading platform when the truck arrives. This usually delays the truck and thus causes further inconvenience to the workers in the following destination warehouses. The office has sometimes received complaints from other warehouses when such a delay in loading means that the arrival times are different from the ones in the truck time-schedule made by the office. For this reason, the office may wish to make a new procedure in the script which enables the truck drivers to record the actual loading and unloading times. For example, a new EDDI table is created for Driver `Gale` to record the actual time of loading and unloading:

```
%eddi
Gale_actualTime (JobNo int, Date char, FromWarehouse char, L_Time_From char, L_Time_To char, ToWare-
      house char, UL_Time_From char, UL_Time_To char);
```

Then a new procedure is created for the office personnel to check and compare the difference between the actual loading/unloading time and the planned ones[21]:

```
%eden
proc check_click : check_mouse_1 {
   if (check_mouse_1[2] == 4) {
      execute("writeln(\"Gale's Proposed Time Schedule\");");
      execute("%eddi\n ?Gale %JobNo, R_Date, FromHouse, L_Time_From, L_Time_To, ToWarehouse,
        UL_Time_From, UL_Time_To");
      execute("writeln(\"Gale's Actual Time Schedule\");");
      execute("%eddi\n ?Gale_actualTime");
   };
};
```

This will display these two tables in the office window interface. For example through the two tables shown in Figure 7.16, the office personnel may find out that the delay of the truck was caused by the delay of both the unloading work at Warehouse London and the loading work at Warehouse Manchester. The auditor may find that this is useful for him to check the daily performance of each warehouse. The manager will also probably support such changes as he can observe the performance of each part of the process in more detail.

Case 4 The customer sometimes complains about the service provided by the warehouse, such as it takes too long time to deal with his requests or sometimes the items are sent to the wrong warehouse. In order to improve this situation, the manager may decide to introduce a bar-code system for identification of items or a system to enable the customers to directly give their requests on-line through

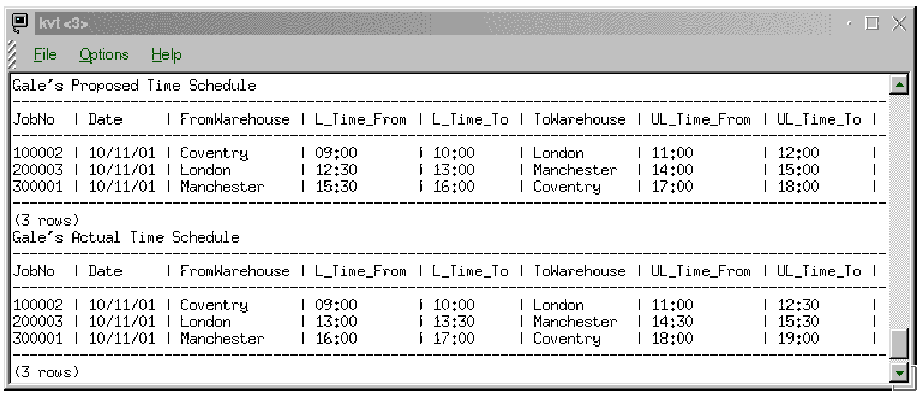

Figure 7.16 Two Tables for the Comparison by the Office Personnel

21.The `execute (string)` function executes a string as Eden statements, where the string must be the valid and complete Eden statement.

the internet. The introduction of such change may narrow the role of the foremen, i.e from that of an active character (i.e. making the redistribution commands) to a passive character (i.e. checking the data received from customers). The involvement of customers in the process of BPR can let them experience the changes and evaluate them. At the same time other personnel in the warehouse may resist such changes because some of them may even lose their jobs if their daily work is automated by the system. The manager in this case can observe their different perceptions about the change and find out the optimal solution for the conflicts.

7.3.5 Concluding Remarks

There is no single ISM which can represent all the aspects of the warehouse state. Typically the state of the warehouse is represented by different ISMs according to what problems are focused on in the SPORE framework. The ISM we construct here incorporates the seed-ISMs for the warehouse: the form-based abstractions that captured the state of the business process model and the activities of the agents; the storage, retrieval and distribution of items; and additional observations such as the one associated with the wider significance of the warehouse operation (e.g. concerned with the legality and the integrity of the business process). The main activities involved in the construction of ISMs in the warehouse case study have been summarised in Sun et al. (1999) that:

- the identification of agents (e.g. foreman, warehouse worker, etc);
- the conception for the roles for these agents corresponding to their characteristic skills;
- the apportioning of responsibilities for particular phases within a given transaction;
- the refinement and formalisation of their precise observables and protocols.

The potential framework for BPR established by applying SPORE can be illustrated by the transformation from a paper-based to a computer-based ISM.

Business process activity can also be viewed from perspectives other than those of the participants within it (i.e. the internal agents), such as the perspectives of an auditor (“What has been going on?” or

"Does it conform to regulations and standards?") and of an analyst ("What could be improved?" or "How could processes be more efficient and stable?"). The distributed version of EDEN (**dtkeden**, see Sun, 1999) enables us to separate the viewpoints of the agents within the model and to complement these with an external observer's interpretation. Figure 7.9 shows how computer-based forms are used to represent the environment for each agent's interaction. In this way, the distributed ISM can serve as a medium by which we can identify and enact appropriate transactions, and refine these through the collaborative interaction between participants. Many possible issues in requirements can be addressed by SPORE in this way such as (Chen et al., 2000a):

- Through experimentation at different workstations (the models), we can identify issues which are problematic from the viewpoint of particular agents: for instance, "how does the office know which drivers are available?", "how does the office determine whether a transaction is completed?".
- Through the elaboration of different seed-ISMs, we can address some additional issues, such as transportation costs, perishable goods, security and trust concerns.
- Through modifying dependencies and communication strategies, we can consider the effects of different technologies, such as are associated with the use of mobile communications, the Internet, optical bar code readers, or electronic locking agents.
- Through collaboration and synthesis of views, we can distinguish between subjective and objective perceptions of state, e.g. to contrast "I remember doing X" with "I have some record of doing X" with "There is an official record of X", or to model misconceptions on the part of an agent.
- Through intervention in the role of superagent, it is possible to examine the consequences of singular conditions which arise from opportunistic interaction or Acts-of-God, and to assess activities outside the scope of normal operation such as are associated with fraud, or manual back-up to automated procedures.