ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Frank Dignum, Jan Dietz, Egon Verharen and Hans Weigand, (Eds)

Communication Modeling - The Language/Action Perspective

Proceedings of the First International Workshop on Communication Modeling, Tilburg, The Netherlands, 1-2 July 1996

The Specification and Implementation of a DEMO Supporting CASE-Tool

Jan L.G. Dietz, Nardo B.J. van der Rijst and Frank L.H. Stollman

Published in collaboration with the
British Computer Society

THE SPECIFICATION AND IMPLEMENTATION OF A DEMO SUPPORTING CASE-TOOL

Jan L.G. Dietz

Department of Information System, Delft University of Technology
P.O. Box 356, 2600 AJ Delft, The Netherlands

Nardo B.J. van der Rijst

Department of Information Systems, Delft University of Technology
P.O. Box 356, 2600 AJ Delft, The Netherlands

Frank L.H. Stollman

Department of Information System, Delft University of Technology
P.O. Box 356, 2600 AJ Delft, The Netherlands

Abstract

In this paper the functional specification of the CASE-tool DEMOcrat is presented, which supports the system analysis method DEMO. DEMOcrat covers four diagramming techniques. First the diagramming techniques are presented in a part of a case study. Next they are formally specified using one of them (the fact model) as the specification language. A prototype version of DEMOcrat is implemented, using Excelerator¹ as the implementation platform.

1 Introduction

The methodology DEMO (Dynamic Essential Modelling of Organisations) has been described and reported upon several times [3-5]. Its practical value has been demonstrated for a few small scale application domains like in [10], and recently in a large scale project [7] as well.

The major motivation behind the development of DEMO was the strongly felt need for information systems analysts to have a theory about organisations (and, by way of analogy, about discrete dynamic systems in general) that is only and purely based on the role of information, taken in a very general sense. Having such a theory at one's disposal would make it possible to understand organisations in a new and original way.

The theory would be independent of other theories, in particular the economic theories about organisations, like the Value Chain Theory [8], or the Transaction Cost Theory [12]. It would have its own value and just be complementary to those other theories. Because of this property, DEMO is most valuable for providing an original informatical contribution to Business Process Redesign and Reengineering, next to and in harmony with contributions from other disciplines, e.g. the economically founded discipline of business administration.

If informatics is understood to be the interdisciplinary field of science that deals with information and communication and their role in the functioning of dynamic systems, especially organisations, then the DEMO theory can most appropriately be called an *informatical* theory.

For some time it was felt necessary to have a CASE-tool at our disposal that supports the specific models and diagramming techniques of DEMO. Since developing such a tool from scratch was beyond the possibilities, we sought for an existing and commercially available tool that could be customised to our needs. Having such a tool would give us the possibility to rapidly build prototypes, that could be tested and commented upon by students from our department. We decided upon Excelerator II which claims to provide extensive customising facilities. It had been made available to us free of charge and it runs under OS/2 and Windows/NT.

The tool that has been designed has got the name DEMOcrat (DEMO conceptualising, refinement and analysis tool). It supports, in an integrated way, the development and maintenance of the four diagramming techniques of DEMO: the communication diagram, the process diagram, the fact diagram, and the action diagram.

The core concepts and the distinguishing features of DEMO are summarised in section 2, in order to let the reader understand the subsequent sections without having to look up other publications. Next, the four diagramming techniques are demonstrated and explained in section 3, using a part of a real world case as a leading example. The

¹ Excelerator is a trade mark of INTERSOLV, Inc. It has been made available to us free of charge by the Dutch company Value Added Consultancy of Amsterdam, the Netherlands.

formal specification of each of them is contained in section 4. The final section 5 deals with various implementation issues.

2. Summary of DEMO

"DEMO" is an acronym for "Dynamic Essential Modelling of Organisations". It is the name of a cross-disciplinary theory about the dynamics of activities in organisations, as well as of an analysis method based on that theory. The disciplines on which it draws are the philosophical branches of semantics and scientific ontology [1], and the social theory grounded in language philosophy [6, 11]. Next to these it incorporates the discrete dynamic system theory as described in [2].

Because a comprehensive description of DEMO in one paper or book is not yet available, we have to refer the reader to fragments of such a description. A relevant set of fragments is constituted by [2-5] and [9]. In the next two subsections a description is provided, as comprehensive as possible, of the aspects of DEMO that are relevant to the scope of this paper.

2.1 The Theory

We will start by explaining the theoretical pillars of DEMO more specifically. The first point to make is that one should profoundly understand that an organisation is a social system. This means that it is composed of social individuals, or *subjects* as we prefer to call them. These subjects influence each others behaviour through communication². In order to abstract from the particular individuals and to concentrate on the behaviour exposed by them, we introduce the notion of actor. So, an *actor* is a particular function or activity to be performed, ultimately by a subject, and an *organisation* is a system of communicating actors.

It is crucial to adopt this basic notion of an organisation fully. Its rightness can easily be demonstrated by imagining that there would be no human beings. Obviously, there would be no organisations either. It is also crucial to recognise that this notion holds for any degree of automation of the activities in an organisation (i.e. of the actors). As we will see in a moment, some activities can easily and without any risk or harm be automated, some however never can. To avoid making mistakes in modelling an organisation it is therefore very helpful to forget for a while about all automation actually present in an organisation and thus to look at it as it was (or would have been) in the pre-automation era.

The actors in an organisation communicate about some world, which we prefer to call the *object world*. It is important to grasp the distinction between the system of communicating actors, or the *subject system* as we prefer to call it, on the one hand, and the object world where the communication is about on the other hand. What constitutes the object world therefore is easily to determine: it encompasses everything where the communication in the subject system is about. Next to the object world, a *system world* is distinguished.

A subject system as well as its corresponding object world are at every moment in a particular state. The state of the system world represents the progress made in performing activities, the state of the object world represents the results of these activities. We come back to these issues after having introduced the concept of transaction.

A well-known distinction in levels of abstraction when studying organisations from the perspective of informatics is the distinction between the documental and the informational level. At the *documental* level an organisation is viewed as a system of actors that produce, store, transport and destroy documents³.

At the *informational* level one abstracts from the substance and the syntactic aspect in order to focus on the semantic aspect of information. What one observes now is a system of actors that emit and receive messages (semantic meanings) to and from each other. This is the level where most current methods and techniques (like e.g. the DFD and the ER-model) aim to be helpful, or, in fact can only be helpful, in spite of farther reaching claims.

It appears to be possible to abstract even further by focusing on the pragmatic meaning of these messages, i.e. on their role in carrying on the business activities. Language philosophy provides the necessary instruments for analysing an organisation at this level of abstraction, which we prefer to call the *essential* level.

What one observes when focusing on the pragmatic aspect is a system in which the actors carry on units of communication⁴ that have a particular effect. We call these units *conversations*. More specifically, they are *performative* conversations, i.e. conversations resulting into an actual change of the state of either the subject system or the object world. We distinguish between two kinds: *actagenic* conversations, resulting in agreements about future

² Communication is to be understood here in its original human sense, not in a technical sense. A good definition therefore would be: sharing mental states.

³ By a document we mean any information carrying substance, like a sheet of paper containing text or a drawing, or a magnetic disk containing bit patterns

⁴ We only take the business communication in an organisation into account. So, chats, expressions of emotions, and conversations not related to the business at hand, are excluded.

actions (agenda of the subject system) and *factagenic* conversations, resulting in the establishment of facts in the object world. Because only in performative conversations, original new things are accomplished, we consider these conversations to represent the essence of an organisation. Furthermore, we call the actions that are agreed upon in actagenic conversations and the results of which are established in factagenic conversations, essential actions, and the conceptualisation of the system observed the *essential model* of the organisation.

Because of the very nature of an organisation, these essential conversations and actions can only be performed by responsible, authorised subjects. Every other activity could as well, and often much more efficiently, be performed by artefacts. This holds especially for all actions that are purely and only informational. There are two kinds of such actions. One kind is the *reproduction* of already existing information, thus the actions usually performed by data base systems. The other kind is the production of information that is only a *derivation* from other information by means of mathematical or logical computation, otherwise said the processing of information.

Figure 1: The Documental, the Informational and the Essential Level of Abstraction

The relationship between the documental, the informational and the essential level of abstraction is depicted in figure 1. This figure must be interpreted as follows. For any organisation there exists at any moment one documental model, one informational model and one essential model. In principle, one may conceive of a number of documental models, all realising the same informational model. Otherwise said, there is a freedom of choice, and consequently a choice problem. Choosing and implementing a documental model is what Information System (Re)engineering is about. The choices are determined by the available information technological possibilities.

Likewise there is a freedom of choice when transferring from the essential level to the informational level. The choice concerns the purely informational actors, i.e. actors that only reproduce or derive information, and the particular messages by which the essential actors communicate in order to carry on their performative conversations. Choosing an informational model is what Information System (Re)design is about. It is part of the more encompassing activity of Business Process (Re)engineering.

Having conceived of the essential level of abstraction and having discovered and understood the crucial role it fulfils in the design of information systems, it is rather astonishing to find that current methodologies do not encompass that level!

The core modelling concept in DEMO is the concept of the (essential) *transaction*. A transaction is considered to be the basic pattern of organisational behaviour. It evolves in three phases: the order phase, the execution phase, and the result phase. Figure 2 exhibits this pattern.

During the *order* phase agreement is reached between actor 1 and actor 2 about the future execution of an action by actor 2. This phase consists of an actagenic conversation, initiated by actor 1, starting at t_1 and ending at t_2 . The result is an agendum (singular of agenda) for the execution of an objective action by actor 2 (an objective action is an action that affects the state of the object world). During the *execution* phase this objective action is executed by actor 2, somewhere between t_2 and t_3 . During the *result* phase actor 1 and actor 2 reach agreement about the facts that have been accomplished as a result of the execution by actor 2. It consists of a factagenic conversation, starting at t_3 and ending at t_4 . Actor 1 is called the *initiator* of the transaction and actor 2 the *executor*.

Figure 2: The Pattern of a Transaction

The behaviour of any organisation thus is conceived as consisting of carrying through transactions. Every (essential) action is embedded in a transaction and every established fact is the result of the successful carrying through of a transaction. This constitutes the DEMO *transaction paradigm*.

Carrying through a transaction can be viewed as a discrete event process, and can thus be modelled by means of a state transition diagram [5]. Every state of a transaction process then is a state of the system world, as opposed to a state of the object world, which is the set of facts established as the result of the successful carrying through of transactions. For example, buying a car or a house is a (transaction) process that may proceed through a large number of distinct subject system states; its successful completion results into the transfer of property which is a fact in the object world.

2.2 The Methodology

On the theoretical foundations explained above, a practical methodology is developed for the informatinal analysis of organisations, in particular for deriving the essential model of an organisation.

Figure 3: The Partial Models of the Essential Model

This essential model is an integrated whole of several partial models, as summarised in figure 3. The communication model contains basically the identified transaction types and the actors that are involved as initiator or executor. The fact model is a specification of the fact types and the constraints that together constitute the state space of the object world. The process model is a specification of the possible transaction processes for every transaction type. The action model is the specification, as far as possible or known, of the behaviour rules of the actors.

The double arrows represent one-to-many relationships. So to one and the same communication model, process model or fact model, a number of action models may belong. Otherwise said, the action model is the core model; every other model can be derived from it. Modifying any of the partial models is considered to be a *redesign* of some business process(es).

The *communication model* of an organisation is the specification of the influencing by the actors of each other behaviour. It is said to represent the interaction structure and the interstriction structure between the actors. By *interaction* structure is understood the mutual influencing through being initiator or executor of transactions. By *interstriction* structure is understood the mutual influencing by means of the subject system and object world state elements that serve as data in the condition part of the behaviour rules that are executed in carrying through transactions.

Every transaction is initiated during the carrying through of some other transaction, specifically during one of the phases of that transaction. Furthermore, proceeding a transaction may have to wait for the progress up to a certain

status of one or more other transactions. The specification of the dependencies in time between transactions constitutes the *process model*.

The *fact model* is the specification of the state space of the object world. It consists of a specification of all relevant fact types, collectively called also the state base, and the specification of all static constraints.

The *action model* is the specification of the rules for every actor that the actor has to follow in performing essential actions, i.e. in conducting actagenic and factagenic conversations, and in executing objective actions.

3. The Diagram Types

There are several diagram types that are used to depict separate aspects of the DEMO essential model. In the next paragraphs the four main diagram types are discussed. In a somewhat simplified form, the Delivery of Leased Lines will serve as an example for all of the diagram types. Therefore, the first paragraph will give a description of this business process.

3.1 An Example: Leased Lines Delivery

The described business process is a simplified version of the business process responsible for the delivery of Leased Lines at the Dutch phone company “PTT Telecom”. This business process has been redesigned using DEMO [7].

The business process starts when the sales representative requests the processing of an order from the order processor. This means that there is a customer order which requests a standard action on a leased line. The standard action might be the first delivery of a leased line, but could also involve moving an endpoint of the connection, or the altering or dissolving of a connection. The order processing begins with the architecture planning, which deals with the necessary co-ordination to make sure the connection can be realised later on in the process.

First the route has to be stipulated, to assert how the connection will be established from one end to the other. This includes the specification of the (customer and phone company) sites that the connection passes through, as well as the material needed at those sites to realise the connection. If it is unknown which material is already present at the customer site, or what the infrastructure is at the customer site, for instance when there are several firms located in one building, a preliminary intake is performed to inform the architecture planner of this data. The preliminary intakes are performed by a mechanic, and since there are other activities that are performed by the mechanics, it is necessary to fine tune the activities needed for the preliminary intakes. This fine tuning is not only concerned with the assignment of the activities to the mechanics within a narrow timespan (of for instance an hour), but has to take preferences of the customer regarding housecalls into account as well.

When the results of the preliminary intake infrastructure are available, the architecture planner can request the allocation of transfer capacity necessary for the connection. This allocation is performed by another part of the phone company, which is not specific for the Leased Lines Delivery process. When the route is stipulated, the order processor can order the material needed at the phone company sites. When the results of the preliminary intake material are available, the material needed at the customer site, can be ordered as well.

Next, the order processor can preplan the activities needed to realise the connection. This involves the high-level planning of the activities within a timespan of several days. The order processor then uses this plan to inform the realisation planner of the work that needs to be done to realise the connection.

The realisation planner first has to fine tune the activities. Like with the preliminary intakes, this involves the planning of the activities needed to realise the connection, again taking the customer’s preferences on timespans into account. When the fine tuning is complete, the transfer capacity is allocated and the necessary material for the phone company site and customer site has been delivered, the realisation planner can request the connecting, installing and fixing of the connection from the mechanics. Afterwards the connection is tested, to check whether all the activities were successful.

If the connection functions properly, the realisation planner reports this to the order processor, who consequently informs the Sales representative that the connection has been established, in other words, the Leased Lines Delivery is complete.

3.2 The Communication Diagram

The communication diagram is a graphical representation of the communication model. It represents the interaction structure and the interstriction structure between the actors. Figure 4 shows the graphical elements of the communication diagram.

Figure 4: The Graphical Elements of the Communication Diagram

An actor is represented by a box, identified by number (A#) and a name. A transaction type is represented by the combination of a disk and a diamond, identified by a number (T#) and a name. For readability purposes, the transaction name is not used in the communication diagram. Instead, the transactions are identified by their number, and the corresponding name and transaction result are given in a separate table: the transaction table. The operational interpretation of the disk is a store for the statuses through which the transactions of that type pass in the course of time. The disk symbol therefore is called a transaction bank. The diamond represents a fact bank containing the facts in the object world that are created as a result of the carrying through of a transaction. These facts are considered to be represented by true propositions stored in the fact bank.

The actors in the system may need to know external fact types, i.e. facts created outside of the system (which is represented by the gray-colored rectangle). The transaction symbol is too specific to represent the external fact types, because the corresponding transaction types fall outside our scope of interest: they are not known. Therefore, the external fact types are arbitrarily put together in external fact banks, which are represented by only the diamond symbol. They are labelled by an external fact bank number (EF#).

The actor who is initiator of a transaction type is connected to the transaction bank by a plain line, the initiator link. The executor of a transaction type is connected to the transaction bank by an arrow pointing to the actor box, the executor link. Interstriction is represented by dashed lines (data links) between actors on one side and fact banks or transaction banks on the other side. As an example, figure 5 shows the communication diagram of the Leased Lines Delivery, with the accompanying transaction table in table 1.

Figure 5: Communication Diagram Leased Lines Delivery

The actors A0, A7, A8 and A9 are external actors, the other actors are internal actors. The transaction types T1, T7, T8 and T9 are boundary transaction types, where T1 is an input transaction type and T7, T8 and T9 are output transaction types, the other transaction types are internal transaction types. The bank EF1 and EF2 are external fact banks.

Transaction type		Transaction result
T1	order processing	The order for the delivery of leased line <connection> has been processed
T2	architecture planning	The architecture of connection <connection> has been planned
T3	route stipulation	The route of connection <connection> has been stipulated
T4	fine tuning	The activities of the mechanics have been fine tuned in fine tuning <fine tuning>
T5	preliminary intaking material	The material <material> present at the customer site <site> is known
T6	preliminary intake infrastructure	The infrastructure <infrastructure> at the customer site <site> is known
T7	transfer capacity allocation	The transfer capacity <capacity> needed for connection <connection> has been allocated
T8	material supply customer site	The material <material> for the customer site <site> has been supplied
T9	material supply phone company site	The material <material> for the phone company site <site> has been supplied
T10	preplanning	The activities needed to realise connection <connection> have been preplanned
T11	realisation planning	Connection <connection> has been realised
T12	connecting / installing / fixing	The necessary connections for connection <connection> have been made / the material for connection <connection> has been installed / connection <connection> has been fixed
T13	testing	Connection <connection> has been tested
External fact banks		Facts present in the external fact bank
EF1	customer data	Material at the customer site Infrastructure of the customer site Customer preferences regarding house-calls by mechanics
EF2	connectiontype test	Information on the test procedure for a certain connection type

Table1: Transaction Table Leased Lines Delivery

3.3 The Process Diagram

The process diagram is the graphical representation of the process model, and is used to specify the transactions in relation to time. With this diagram it is possible to depict the three phases of every transaction, the Order, Execution and Result phase (O,E,R), and the initiations of a transaction from within another one. Input transaction types are initiated externally, kernel and output transaction types are initiated within a transaction phase. The graphical elements of the process diagram are given in figure 6.

Figure 6: The Graphical Elements of the Process Ddiagram

A transaction that is not decomposed into its three phases is depicted with a thick circle. The completion of the order phase of a transaction always leads to the execution phase, which in turn, after completion, brings the result phase about, this is represented by the initiation link between them. The dashed arrow depicts a completion condition, the process at the start of the arrow has to be completed before the process or initiation at the arrowhead can be completed. Whenever the word process is mentioned, this is either a transaction phase or an undivided transaction. As an example, figure 7 shows the process diagram of the Leased Lines Delivery.

Figure 7: Process Diagram Leased Lines Delivery

Only transaction T1 is initiated externally, which is consistent with the communication diagram, as T1 is the only input transaction type. Note that, although transaction types T7, T8 and T9 are output transaction types, so no known transaction types are initiated from within one of their phases, they are still divided into the three phases. This has to do with parallelisation of the transaction types. The transaction types are initiated as early as possible in the business process, but the results are not needed right away. To depict this, there are two condition arrows from the transaction type to another. One from the order phase back to the initiating process (the transaction has to be promised) and one from the result phase to the initiation point of the process that needs the results from these transaction types.

3.4 The Fact Diagram

The fact diagram is the graphical representation of the fact model. A fact model in DEMO is basically the specification of a first order sorted logic language. Instead of the conventional Peano-Rusell notation however a graphical notation is used, called the fact diagram. The graphical elements of the fact diagram are given in figure 8.

Figure 8: The Graphical Elements of the Fact Diagram

The predicates are called *fact types*. They may be of any arity and are identified by a predicate number (FT#). The graphical notation of a fact type is a sequence of adjacent rectangles, as many as the arity of the fact type. E.g., the graphical notation of a binary fact type thus consists of two adjacent boxes (see figure 8). The meaning of the fact type is expressed in the natural language sentence type printed in italics under the sequence of boxes. The brackets “<” and “>” are meta-brackets. The sentence “<2> is of connection type <1>” should be interpreted as: “the object in role 2 is of connection type of the object in role 1”.

The sorts are the extensions of unary fact types, called *object classes*; they are represented in a fact diagram by circles or ellipses. The circle represents the set of objects that are referent in the fact type or the class.

Every role (of every fact type) is linked to an object class, called its reference class or *domain*. Exceptions to this rule form the so-called *primal entity types*. The extensions of these fact types are the basic sorts of the logic language. They represent either a scale, depicted by a circle with a letter, or a medium, type depicted by a bold circle with a letter. In this paper we use the absolute (A) and categorical (C) scale, and the text (T) and formula (F) media. There is also a so-called *derived fact type*, depicted by hatching of the fact type. The extensions of derived fact types are not created but inferred or computed.

Figure 9: Fact Diagram of the Leased Lines Delivery

There are several types of constraints which can be expressed in a fact diagram, like a *uniqueness constraint*, which is depicted by a line above the role-box, representing that every fact type from the referred object class can occur at most once as referent in this role. Other types of constraints are the *exclusion constraint*, which is depicted by a dashed line with an arrow pointing to the role at each end of the line, indicating that the same object cannot occur in both roles, and the *mandatory constraint*, depicted by a dot at the object class side of the line, indicating that every object in that class has to play the role at the other end of the line at least once. As an example figure 9 shows the fact diagram of the Leased Lines Delivery.

The arguments of an atom, e.g. FT13: is_of_connectiontype_test(a,b), are called the *referents* of the corresponding fact; they are *objects* in the object world of the system described [3]. These objects play different *roles* in the fact, which are indicated by the numbers "1" and "2" in the boxes. These numbers thus are relative role numbers.

3.5 The Action Diagram

The action diagram is the graphical representation of the action model. The action model specifies the behaviour rules of the actors from the communication diagram. This is depicted by an action diagram per actor per transaction phase. The representation is based on the standard ANSI flowchart. The graphical elements of the action diagram are shown in figure 10.

Figure 10: The Graphical Elements of the Action Diagram

Although the action diagram allows the description of the informational and/or documental support corresponding to each essential action, we will not use this extended notation. Since we only describe the essential model of our example we will limit the diagram to the essential actions. As an example figure 11 gives the action model of the order phase of T1, “order processing”, which is executed by A1, “process order” from the Leased Lines Delivery.

Figure11: Action Diagram T1/O of Leased Lines Delivery

As one can see, the order phase of this transaction involves quite some communication. First the architecture planning is requested, then some facts are needed before all the material needed for the connection (which is determined during the route stipulation) can be requested (what is already present at the customer site). Only after the delivery of the material has been promised (and thus the route is stipulated and the preliminary intakes are finished) will the order processor promise the processing of the order of the Leased Lines Delivery.

4. The Formal Specification of the Diagram Types

To formally specify the diagram types of DEMO, the methodology DEMO was used itself. To do this, meta-models of the existing models were made, using the fact diagram to depict these meta-models. However, there was one constraint: since Excelerator II was going to be used to implement DEMOcrat, the maximum arity of the fact types in the formal specification is two, which means that only unary and binary fact types can be used.

Figure13: The Fact Diagram of the Process Diagram

One remark is in order regarding the fact diagram of the process diagram. The object class transaction phase consists of the order, execution and result phase of the transaction types, e.g. T1/O is an element of this object class.

Figure 14: The Fact Diagram of the Fact Diagram

Two remarks regarding the fact diagram of the fact diagram are in order: The dashed circle, labelled *Fact type#* is a so called *identification class*, used here to identify fact types by a fact type number. Note that *Unitype* is in fact short for *Unary Fact type*.

Figure 15: The Fact Diagram of the Action Diagram

The specification of the action diagram is in fact more a specification of the possible elements of the action diagram, than it is a specification which relates the action model to the other three models. The result is that it is only possible to check whether an action diagram is syntactically correct, but not whether it is correct in relation to the other diagram types.

5. From Specification to Implementation

Excelerator II supports a wide variety of system analysis methods. Users of this CASE-tool can adapt the tool to their specific needs by means of the Customizer⁵. This customising facility was used by us for the development of the first prototypes of DEMOcrat.

One of the main problems we were confronted with, was the implementation of the specifications of DEMOcrat in Excelerator. Excelerator's information model is based on the binary entity-relationship model. Therefore the translation of the fact diagrams was only possible when no fact types with an arity greater than two were involved. However, this did not turn out to be an insurmountable obstacle, and the specifications as presented in the previous paragraph could be mapped fully on the modelling formalisms of Excelerator.

Two different modes of specification are present when using Excelerator, the graphical mode and the dialogue mode. In the *graphical* mode the analyst can choose from the different *graphical* object types to draw the diagrams. The objects and their relationships are instances of the fact types defined in the specifications. In the case of the communication diagram e.g., the analyst starts with the identification of the system (and consequently of the environment). Next, the actors in the environment and the system are specified. Then the transaction types can be introduced. For each transaction type one has to specify the initiator as well as the executor. After these steps, that represent the interaction structure, the interstriction between the different actors can be specified by drawing the dashed lines as a representation of informative conversations.

In the *dialogue* mode all input is provided by means of dialogue boxes. The specification process in the dialogue mode follows the same sequence of steps as in the graphical mode. The advantage of working in the dialogue mode is, that the interactive manipulation of the data ensures correct storage in the underlying database.

In both modes the input data are stored as entries in a relational database. Testing of the prototype by several students, however, showed that the *dialogue* mode is in fact the only way to ensure correct data storage. E.g. in the more intuitive *graphical* mode, deletion of an object will not guarantee that it, and its relations, will be removed from the underlying database. To ensure correct storage, Excelerator advocates the obligatory use of the *dialogue* mode.

As mentioned in the introduction, Excelerator II only runs under OS/2 and Windows/NT. This means that diagrams have to be converted before they can be used in a document running under for instance Windows 3.xx or Windows 95. The quality of the converted diagrams turned out to be very poor. For a reasonable result, diagrams had to be redrawn in for instance a flow-chart application, as was used for the diagrams in this article.

The results of tests lead to the conclusion that, although Excelerator II provides extensive customising facilities, it is not suitable as the implementation platform for the 'commercial' version of DEMOcrat. A suitable implementation platform for DEMOcrat has not been found yet.

References

1. Bunge M.A. Treatise on Basic Philosophy, Vol. 3 and Vol. 4, D. Reidel Publishing Company, Dordrecht, The Netherlands, 1979
2. Dietz J L G: A Communication Oriented Approach to Conceptual Modelling of Information Systems. In: Lecture Notes in Computer Science 436, Springer Verlag, Berlin, 1990, pp 441-460
3. Dietz J L G: Subject-Oriented Modelling of Open Active Systems. In: E.D. Falkenberg, C. Rolland, E.N. El-Sayed (eds.), Information Systems Concepts: Improving the Understanding. IFIP Transactions A-4, North Holland, Amsterdam, 1992, pp 227-238
4. Dietz J L G: Business Modelling for Business Redesign, In: Proceedings of the 27th Hawaii International Conference on System Sciences, IEEE Computer Society Press, Los Alamitos, CA, 1994a, pp 723-732
5. Dietz J L G: Modelling Business Processes for the Purpose of Redesign. In: Proceedings IFIP TC8 Open Conference on BPR, Australia, North-Holland, Amsterdam, 1994b, pp 249-258
6. Habermas, J. Theorie des kommunikativen Handelns, Erster band, Suhrkamp Verlag, Frankfurt am main, 1981

⁵ Customizer is a trade mark of INTERSOLV, Inc.

7. Jansen M, Poot F C J: DEMO: a method for BPR? (DEMO: een methode voor BPR?). Master thesis Department of Information Systems, Delft University of Technology, Delft, 1995 (in Dutch)
8. Porter M.E. Competitive Advantage, Creating and Sustaining superior performance, The Free Press, New York, 1985
9. van Reijswoud V E, van der Rijst N B J: Modelling Business Communication as a Foundation for Business Proces Redesign: A case of production logistics. In: Proceedings of the 28th Hawaii International Conference on Systems Sciences. IEEE Computer Society Press, Los Alamitos CA, 1995, pp 841-850
10. van der Rijst N B J, Dietz J L G: Modelling the Essential activities in Job Shop Planning and Scheduling for Decision Support System Development. In: Proceedings of the 1993 European Simulation Symposium -ESS93, Society for Computer Simulation, San Diego, 1993, pp 198-203.
11. Searle J.R. Speech Acts, an Essay in the Philosophy of Language, Cambridge University Press, Cambridge, 1969
12. Williamson O.E. The Economic Institutions of Capitalism, The Free Press, New York, 1985