Expert in Disaster Recovery Scenarios

Michel Verheijen and Marcel E.M. Spruit

Many organizations rely heavily on the availability of their information systems. It is the responsibility of the disaster recovery discipline to provide continuity, even after the occurrence of a disaster. To achieve this, disaster recovery makes use of disaster recovery scenarios. Existing methodologies for setting up such scenarios lack proper support. As a result of that only experts can set up the scenarios. However, in this paper we show that expert system techniques can play an important role in setting up disaster recovery scenarios.

1. Introduction

The availability of the information systems is an important topic, since most organizations have become increasingly dependent on information systems. The dependence can be such that the continuity of an organization depends completely on the availability of its information systems. Protecting the availability of information systems against a variety of threats is the scope of information security. This discipline has a twofold objective: to prevent the occurrence of a security breach as a result of threats, and to restrict the damage after the occurrence of a security breach. A special area of attention of the latter is disaster recovery. Disaster recovery aims for continuing the operation of computer systems and networks after a disaster has taken place. This means that computer and network operations will fall back to other facilities, when the original equipment is not available anymore.

To achieve an effective disaster recovery, it is necessary to set up a disaster recovery plan beforehand. In general such a plan consists of two parts:

1. A detailed description of the arrangements made in advance, such as backup procedures, special resources, hardware, software, data-communication and a recovery location.

2. A disaster recovery scenario, describing the activities to be executed in case of a disaster.

There exist several (commercial) methodologies for setting up these parts, but all existing methodologies lack proper support for setting up a disaster recovery scenario. Consequently the development of an adequate scenario depends completely on the knowledge and experience of its designer. Except for some general guidelines and examples, no support is offered by the methodologies and tools.

The objective of this article is to describe a computer application, which supports the setting up of a disaster recovery scenario, based on knowledge, which has been put into the application. Generally this is realized by using an expert system. This disaster recovery expert system should offer the following functionality:

- Knowledge on disaster recovery can be put into the system.
- Knowledge can be added, modified and removed.
- Knowledge, stored in the system, can be reproduced.
- The system can reason on the basis of its knowledge.
- Reasoning of the system is traceable.
- The system can generate a framework of a disaster recovery scenario.
- The system can generate solutions related to specific disaster recovery circumstances.

To come to such a system, we first analyzed how to set up a disaster recovery scenario in practice. Next we examined the relevant expert system techniques and how to apply the techniques to support the setting up of a disaster recovery scenario. To be able to develop the system, we made a functional design and we described the knowledge, which has to be put in. Finally, we developed a prototype to determine the benefits in practice of a disaster recovery expert system.

2. Disaster recovery scenario

A disaster recovery scenario describes all activities, which have to be carried out after some kind of disaster occurred (like a fire, an earthquake, severe technical problems with the computer system, etc.). The first part of the scenario describes the escalation process, which leads to a decision whether to fall back or not. Falling back means that the company will start using computer systems on another location to be able to continue its business process(es).

2.1 A Framework of a disaster recovery scenario

Usually a disaster recovery scenario is set up in co-operation with a consultant of a disaster recovery center. In general the resulting scenario consists of a number of milestones and main components. The main components are completed with activities. This framework of milestones and main components is shown in Figure 1. A dashed line indicates that the main component is not always included in the scenario.

Execution of a disaster recovery scenario is initiated by a disaster. During the escalation process the so-called crisis team decides either to fall back or not. This process can be included in the disaster recovery scenario, but sometimes it is already described in emergency procedures. If one decides to fall back, people on the recovery location have to be notified. Parallel to this, everyone concerned can be informed and the teams defined in the scenario can be called.

Sometimes people are informed by another team than the crisis team. In this case the team that informs has to be called first. That is why there is a dashed arrow between ‘calling teams’ and ‘informing’. Informing should be a continuous activity. At any time people concerned should be able to get answers on their questions about the current situation. An important aspect of informing is the notification of reaching a milestone. These notifications are essential for a smooth progress of the fall back.

After the recovery teams have been called, a number of activities can start in parallel: the computer systems and datacommunication facilities can be prepared, a temporary workplace has to be fit up and one can start with making an inventory of lost data. Remaining activities, like fitting up a crisis center, engaging temporary employees and arranging overnight stays, may also be performed.

After the computer systems, including the datacommunication, has been prepared, it should be tested. When the test is successful, the system can be released. The disaster recovery arrangement is operational when the system is released and the temporary workplace is ready for use. The last phase of the disaster recovery consists of recovering the lost data.

2.2 Setting up a disaster recovery scenario

Usually disaster recovery centers use a methodology to set up disaster recovery scenarios. Such a methodology contains the following phases:

1. Preparation.

This phase generally consists of three subphases:

a) Specification of conditions.

   The conditions define the scope of the disaster recovery scenario. One can think about:

   - The maximum period of time to make the disaster recovery arrangement operational. Within this time all activities between the milestones "Disaster" and "DR arrangement operational" (Figure 1) should be completed.
   - The worst case situation, for which the scenario will be designed.
   - The loss of data, which can be recovered by the users. This defines the required frequency of making backups and putting them in safety.
   - The critical applications and their priorities. Critical applications should be restored after a disaster. The order in which they should be restored is given by their priorities.
   - The needs of the users to continue their work.

b) Description of disaster recovery arrangement.

   This concerns the description of measures and resources, which should be arranged in advance. Usually this is not described in the disaster recovery scenario, but in the first part of the disaster recovery plan (see section 0). The following subjects are dealt with:

   - Hardware and software required at the recovery location.
   - Data-communication required to connect users to the computer systems in a fall-back situation.
   - Procedures for making backups adequately.
   - Special resources needed during the normal business processes, like special forms, labels, etc.

c) Description of disaster recovery organization.

   This concerns the allocation and description of teams, which perform the disaster recovery activities. To generalize the description, team members are indicated by function rather than by name.

2. Brainstorm sessions.

The disaster recovery scenario is set up during brainstorm sessions. Activities and their mutual relations are determined and teams, resources and timings are assigned.

Usually one starts with the worst case situation and tries to go step by step through the whole disaster recovery process. The results of the preparation phase and existing procedures (e.g. emergency, backup and telephone procedures) are taken into account. The first main component of the scenario is the escalation process. If this is already described in emergency procedures, it can be omitted. Otherwise the necessary activities of the escalation process are searched for by answering questions like for example:

- Who is contacted, when a user finds out that the computer system doesn't work anymore?
- Who is informed and who is called? By whom? Will there be a special helpdesk?
- What happens if, after a first analysis, one still doesn't know how to solve the problems?
- What are the possibilities to be able to continue working?
- Is a backup made of the latest data?
- What else need to be done before an actual fall-back?

The last activity in the escalation process is to decide whether to fall back or not. If one decides to fall back, the rest of the scenario becomes relevant. The necessary activities in this part can be determined by answering questions like for example:

- Who is informed and called after the decision to fall back?
- Is it necessary to equip a crisis center to manage the recovery operation?
- What resources do you need at the recovery location and how do they get there (think also about people and backup tapes)?
- Which restore activities (for computer systems, applications and data) are necessary?
- Do you have batch jobs during normal business processes that result in extra activities in a fall back situation?
- How will the systems be tested?
- How are connections for datacommunication set up and how are they tested?
- Is it necessary to equip a temporary workplace? How do the users get there?
- Are there other (less critical) activities, which should be included in the disaster recovery scenario (like engaging temporary employees, arranging overnight stays, distributing print-output)?

Relations between activities and assignments of teams, resources and timings are generally determined by using common sense and experience.

3. Checking conditions and solving problems.
After the brainstorm sessions one has to check whether the scenario complies with the conditions specified in phase 1. Problems have to be solved, for example by splitting activities in several steps, which can be executed in parallel, by removing activities, which can be prepared in advance and by moving non-critical activities from the critical path. If all conditions are fulfilled, one could return to phase 2 to have brainstorm sessions for no-worst-case situations.

4. Implementation of disaster recovery scenario.
Finally the scenario has to be implemented. This means implementing changes in the daily procedures, detailing the activities in the disaster recovery scenario, creating automatic procedures (e.g. for making and restoring backups), doing a simulated run and testing (parts of) the scenario in several situations.

2.3 A supporting computer application

A computer application may be used to support the setting up of a disaster recovery scenario. In the first place such an application should generate the milestones and main components, as discussed before. Subsequently the main components should be completed with relevant activities and relations between activities. Moreover the activities should be assigned to teams.

To get such an application, the necessary knowledge about disaster recovery has to be specified and put in. By reasoning with the knowledge, the application can generate the framework of a disaster recovery scenario. Generally this is realized by using an expert system.

3. Expert systems

3.1 What is an expert system?

Intelligence is often considered as being able to collect knowledge and to reason with this knowledge in order to solve (a certain class of) problems. Already for decades the possibility to catch intelligence in a computer is investigated in the field of Artificial

Intelligence. Artificial Intelligence can be defined as "*the science of making machines do things that require intelligence if done by men*" [1].

An expert system is an application of Artificial Intelligence. Using a computer program one tries to imitate the ability of an expert to solve problems. This means that an expert system is a computer program that can solve problems by making use of knowledge that has been put into its database.

The main components of an expert system are the knowledge base and the reasoning component. The knowledge base is a database, storing all the knowledge needed for solving the kind of problems the expert system has been designed for. The reasoning component consists of an engine that uses the knowledge in the knowledge base and the input from the user to draw conclusions. The method of reasoning used by the expert system determines the way this engine works and the way knowledge is represented in the knowledge base. Three basic methods of reasoning are discussed in the next section.

We distinguish between two approaches to implement a program, which can solve problems by reasoning with knowledge: implementation as a conventional program [1], or as an expert system. A conventional program executes its instructions sequentially (taking into account that, depending on the value of certain variables, some parts of the program are executed and some are not), while the order in which steps are taken in an expert system is not determined in advance but at run-time by the reasoning engine. The latter has the disadvantage that it is difficult to understand how the system works, which makes finding bugs more complex and time-consuming. On the other hand, an expert system has several advantages compared to a conventional program:

- Flexibility of knowledge base. Knowledge in the knowledge base can be adapted and extended easily, while changing knowledge in a conventional program means changing the program (instructions) itself.

- Accessibility of knowledge base. As the knowledge base is separate, it is better accessible, both for the expert system as for the user. This makes it easier to reproduce the stored knowledge.

[1] A conventional program is computer program which consists mainly of algorithms. In such a program knowledge can be inserted by using 'normal' instructions.

- Reusability. If the knowledge is separated strictly from the rest of the program, it is possible to reuse the program to solve similar problems. For example, the disaster recovery expert system may be reused to develop expert systems for setting up different kinds of scenarios.

- Existence of expert system shells. Expert system shells provide a complete environment for developing expert systems. The basic parts of the knowledge base and the reasoning component are implemented already and by using several interactive tools an expert system can be developed within a relatively small period of time. Besides, using an expert system shell generally leads to a well maintainable system.

3.2 Expert system techniques

To solve a problem an expert system uses a specific method of reasoning. In general three different methods of reasoning can be distinguished [2]: Rule-based Reasoning, Case-based Reasoning and Constraint-based Reasoning. Each will be discussed briefly.

Rule-based Reasoning

A system which is based on Rule-based Reasoning, reasons on the basis of *rules* and *facts*, stored in the expert system database. Rules are used to define relations between facts and possible conclusions. Usually rules have the form '*if* 〈CONDITION〉 *then* 〈CONCLUSION(S) / ACTION(S)〉'. Various conditions can be combined into one new condition by using logical operators (like 'and' and 'or'). Stored rules and facts are used by the *inference engine* to come to new facts or conclusions. Two different mechanisms can be used in the inference engine: forward chaining and backward chaining.

Forward chaining (see Figure 2 [3]) is based on the recognition of a situation. The rules are processed one by one. If there is a rule for which all conditions in the IF-part (the antecedent) are met, then the THEN-part (the consequent) will be executed (the rule 'fires'). Usually this results in one or more new facts. This process continues until no rule can fire anymore.

Backward chaining (see Figure 3 [3]) starts from a goal. The mechanism tries to prove a consequent. First it is checked if the consequent is already present as a fact in the database. If this is not the case, all rules which can lead to the consequent are collected. A rule can only lead to the consequent, if all conditions in the antecedent of the rule are met. The next step is to try to prove all conditions.

Case-based Reasoning

Case-based Reasoning is based on the principle that people use analogies and examples during thinking and reasoning. The essence of Case-based Reasoning is that knowledge is represented as cases. Each case is characterized by a set of specific values, like the old problem situation, the solution applied and the procedure, which led to the solution. New problems are solved by considering a previous, similar situation and by reusing the solution or the procedure applied at that situation. The solution is evaluated and, if necessary, adjusted somewhat to fit the new situation. The next step is the learning process in which experiences, gained during solving the problem, are stored. The Case-based Reasoning process is shown in Figure 4.

Constraint-based Reasoning

Sometimes a problem can be defined by a set of constraints. Solving such a problem is finding values of variables so that all constraints are met. Generally it is not possible to find the solutions by considering all possible combinations of values. Instead one has to start with reducing the search space, for example by using a consistency technique. Such a technique reduces the number of possibilities by eliminating combinations of values, which cannot appear together.

Several Constraint-based Reasoning techniques exist. Examples are:

- The ‘Generate and test’ algorithm generates a possible solution. If this solution is a good solution (which means that all constraints are met), the algorithm stops. If not, the solution is modified and tested again. The algorithm continues until a good solution has been found.
- The ‘Chronological backtracking’ algorithm proposes values of variables in a certain order. If the generated solution with a new variable value is still valid, a value is proposed for a next variable. If the solution is no longer valid, one goes one or more steps back (‘backtracking’) to try to get a valid solution by assigning other values to variables.

- ‘Dependency-directed backtracking’ works almost the same as ‘Chronological backtracking’. The only difference is that, in case of backtracking, the value of the most recent variable is not changed. Instead, the value of another variable is changed, namely a variable, which is closer to the cause of the constraint conflict. In this way unnecessary backtracking is avoided.

3.3 Relevant expert system techniques

We have to determine which expert system techniques are appropriate for setting up a disaster recovery scenario.

Rule-based Reasoning

Rule-based Reasoning is a technique, which can be used in many cases. The knowledge has to be described in terms of rules and facts. It is likely that this will not cause any problems for the knowledge.

Case-based Reasoning

Case-based Reasoning requires a case-base containing a large number of solutions. This means that a large number of disaster recovery scenarios is needed. This is very hard to accomplish. Some reasons are:

- Creating a good accessible and sufficiently extensive case-base costs quite a lot of time and money.

- It is difficult (or even impossible) to classify the existing disaster recovery scenarios according to problem characteristics. This is caused by the fact that the decision to include or exclude activities, as well as the level of detail that is necessary, depend more on personal preferences than on company characteristics.

- If, for a certain company, there is no proper case in the case-base, it is not possible to choose a good solution. Consequently, the user has to spend a lot of time on tailoring the generated solution to the actual situation. It is even likely that the effort does not lead to a suitable disaster recovery scenario at all.

- As disaster recovery scenarios are confidential, most companies do not allow to include their scenario in a (public) case-base.

For these reasons, Case-based Reasoning will not be suitable.

Constraint-based Reasoning

Constraint-based Reasoning requires that each problem is defined by a set of constraints, which the problem has to meet. An example of such a problem is setting up a timetable for a school. This problem is characterized by a large number of possible solutions while only a couple of them meet the defined constraints (e.g. a certain group cannot have different lessons at the same time, the number of intermediate hours should not be more than three, a teacher can only give one lesson at a time, etc.). However, the problem of setting up a disaster recovery scenario is not characterized by a set of constraints on the basis of which a good scenario can be accomplished. For this reason, Constraint-based Reasoning will not be suitable.

Conclusion

Considering the arguments mentioned above, only Rule-based Reasoning can be used in an expert system that supports the setting up of a disaster recovery scenario. In the following section we explain how to apply this expert system technique.

4. Applying Rule-based Reasoning

4.1 Generating the framework

In section 2.1 we described a framework of a disaster recovery scenario. Such a framework can be generated by using the expert system technique Rule-based Reasoning. As this technique uses rules for reasoning, we have to define the rules.

Generating the framework can be subdivided into four steps:

1. Generating the milestones and main components.

2. Elaborating the main components into activities.

3. Defining the relations between the activities and milestones.

4. Assigning the activities (and milestones) to teams.

Generating the milestones and main components

The milestones and main components to be generated can be found straight on in section 2.1.

Elaborating the main components into activities

The main components, defined in the previous step, can be elaborated by using the questions, which are usually asked during the brainstorm sessions as described in section 2.2. On the basis of the answers the relevant activities can be determined. Therefore, a set of rules has to be defined for each main component. Each rule (used for Rule-based Reasoning) has the form '*if the answer on question* ⟨QUESTION⟩ = ⟨ANSWER⟩ *then include* ⟨ACTIVITY⟩ *in the scenario*'. For getting the answers, it is necessary that the expert system asks the relevant questions to the user.

Rule-based Reasoning is based on rules, which include specified answers. Consequently, each question has a limited number of answers, which comply with the corresponding rule(s). Only those answers need to be recognized. Answers given in natural language cannot be recognized by the expert system, as it neither knows nor recognizes natural language. Therefore, the expert system has to present each question including the possible answers.

While studying a number of existing disaster recovery scenarios, a lot of similarities showed up. In fact, the rough framework of each disaster recovery scenario can be built up by selecting activities from a particular set of activities. Some of those activities were even present in every scenario. Knowing this particular set of activities reduces the number and the complexity of questions to be asked for.

Defining the relations between the activities and milestones

After studying a number of existing disaster recovery scenarios, it appeared that we can distinguish between two kind of relations:

- Relations that are fixed for given activities. An example is 'the final notification of fall-back' which always follows 'the decision to fall-back'. Hence both activities have a fixed relation.

- Relations that depend on the activities as well as the specific situation. This can be illustrated by the activity ‘transportation of data-communication equipment’. If a team of the company will carry out the transport, then the team has to be called first. This implies a relation between calling the team and transporting the equipment. On the other hand, if another party takes care of the transport, another relation will be defined instead.

Fixed relations can be put as facts in the expert system database. Based on a rule, these relations can be generated by the system automatically, so without intervention of the user. The rule will fire if both activities, which make up a fixed relation, are included in the scenario.

Relations, which are not fixed, depend on certain conditions. To determine the value of the conditions, questions have to be asked to the user. Each condition can be included into the if-part of a rule, so that it is possible to generate the relation on the moment that the value of the condition is known.

Assigning the activities (and milestones) to teams

For each activity one has to specify a team which will perform the activity. From existing scenarios it appeared that there are teams, which are generally present in any disaster recovery scenario. It also appeared that there are particular sets of activities, which are always assigned to a particular team. An exception is the set of activities, which are part of the escalation process, because these highly depend on the other company’s processes.

Teams that are generally included in a scenario, can be put as facts into the expert system database. By defining a rule, which fires when an activity is included, the activity can be assigned to a team automatically.

The assignment of milestones to teams is performed similarly.

4.2 Activities for specific computer systems

The activities in the main component ‘preparing computer systems’ can be specified in general, which means that they are valid for all types of computer system. Examples of such activities are ‘restore the operating system’ and ‘test the applications’.

In practice only a limited number of different computer systems is used very frequently. Therefore, it is possible to include in the expert system specific (and thus detailed) activities for those systems. As soon as one knows which computer system is used, the appropriate activities for this system can be generated by the firing of a rule.

4.3 Transport of resources

During a disaster recovery operation, activities take place on several geographical locations (e.g. the locations of the company, the recovery arrangement, the backup media and the temporary workplace). To transport resources, which are needed on other locations, (logistic) processes have to be performed. These processes have to be included in the disaster recovery scenario as (transport) activities.

For every activity in the disaster recovery scenario, it has to be specified, which resources are needed to perform the activity. It also has to be known on which location the activity will be performed and on which location the resources are available. If this is known, the expert system can generate the necessary transport activities. The expert system can also define a relation between the transport activity and the activity, which needs the resource. If several resources on the same geographical location will need transport to the same destination, then the associated transports can be combined into one activity. These mechanisms can all be included in the expert system by means of rules.

5. A functional design

A functional design describes *what* the expert system should do. The functional design will be set up using the Object Modeling Technique (OMT) (see [4]).

5.1 The Object Modeling Technique

The Object Modeling Technique is an object-oriented technique: the emphasis is on *classes* and *objects*. Objects are discrete, distinguishable entities, each of which has its own identity, like a blue ballpoint or John’s car. Objects with the same data structure and behavior are grouped into a class. A class is an abstraction that describes the properties (attributes) which are relevant to an application. An object is called an instance of its class. For example, the object ⟨Ferrari, red, AHL-4560⟩ can be an instance of the class Car.

The last concept discussed is inheritance. Inheritance is based on the hierarchical relationship between classes: a subclass incorporates, or inherits, all of the properties of its superclass and adds its own unique properties. An example is a class Line, which inherits the attributes ‘color’ and ‘width’ from a class Figure and adds its own attribute ‘length’ and its own method ‘draw’.

The Object Modeling Technique uses three related models to describe a system:

- The <u>Object Model</u> describes the classes and their relations. This model is the most essential, because it describes *what* is changing or transforming.
- The <u>Dynamic Model</u> describes the behavior of the classes: the states in which the objects can be and the possible transitions between states (caused by events).
- The <u>Functional Model</u> describes the data flows and the corresponding processes and functions.

5.2 A Sketch of the Object Model

To get an impression of the Functional Design of the expert system, we present a sketch of the most essential model: the Object Model. The Object Model is based on the description of the expert system as given in section 2.3.

The system will be a rule-based expert system. Therefore we can distinguish an inference engine, rules and facts (see section 3.2). Facts are related to teams, questions, scenario elements, instructions, relations and resources. We can consider all these items as different classes. Finally, there is a class for the user. All classes and their relations are shown in the Object Model (Figure 5, and legend in Figure 6). A brief description of the classes is given below.

1. *Class Inference Engine*

The inference engine takes care of firing rules in case their conditions are met.

2. *Class Rule*

A rule defines a relation between one or more facts and a possible conclusion or action.

3. *Class User*

The user is the person who uses the expert system to set up a disaster recovery scenario.

4. *Class Team*

A team is a group of one or more persons, responsible for an activity.

5. *Class Question*

A question is a request for information to determine which activities should be part of a particular disaster recovery scenario.

6. *Class Scenario Element*

A scenario consists of a couple of related elements. A scenario element of a disaster recovery scenario is a milestone, or an activity. An activity can be a transport or another activity.

7. *Class Instruction*

A scenario element (an activity) can be divided into a number of (detailed) sub-elements, called instructions. The instructions describe the steps to be taken to complete the activity.

8. *Class Relation*

A relation between scenario elements defines the order in which the elements (milestones or activities) should be handled.

9. *Class Resource*

Some activities require one or more resources. Either a resource already exists, or it is produced by another activity.

Instances of the classes Scenario Element (Activity, Transport and Milestone), Question, Team, Relation, Instruction and Resource make up the facts in the expert system database. Relations between the facts and possible conclusions or actions are defined by instances of the class Rule. That is why there are associations (relations) between the class Rule and all other classes mentioned before. The facts and rules will be used by the inference engine to come to new facts or conclusions. The user controls (starts and stops) the inference engine, receives messages and answers questions.

6. Inserting knowledge

6.1 Facts

As mentioned before, instances of the classes Scenario Element (Activity, Transport and Milestone), Question, Team, Relation, Instruction and Resource make up the facts in the expert system database.

As discussed in section 4, facts can be found by studying existing disaster recover scenarios and by analyzing brainstorm sessions (see section 2.2). The set of found facts, except for transports, can be put into the expert system database. Transports need not be put in the database, as they can be generated by using rules (see section 4.3 and the logistic rules described in the next section). Initially, all facts have status 'undefined', which means that they are not yet included in the disaster recovery scenario. By firing rules, facts may get status 'defined' and become part of the scenario.

6.2 Rules

The expert system will use rule-based reasoning as described in section 4. The expert system should contain several types of rules:

1. rules for asking questions

Depending on the answers on questions, some activities are included in the scenario, while other activities are not. To obtain the answers from the user of the expert system, the questions have to be asked by the firing of rules.

2. rules for defining scenario elements [2]

Some scenario elements will be part of every disaster recovery scenario. They can be generated by the firing of a rule:

'*if* ⟨SCENARIO ELEMENT⟩ *is undefined*

then *define* ⟨SCENARIO ELEMENT⟩'.

For scenario elements which depend upon the answers on questions, rules of the following form can be specified:

'*if* ⟨SCENARIO ELEMENT⟩ *is undefined*

and *the answer on question* ⟨QUESTION⟩ = ⟨ANSWER⟩

then *define* ⟨SCENARIO ELEMENT⟩'.

3. a rule for defining instructions

Instructions describe the steps, which have to be taken to complete an activity. Every instruction belongs uniquely to one scenario element. This means that instructions can be defined as soon as the scenario element, to which they belong, has been defined. This can be realized by the rule:

```
'if   ∃ scenario element s which is defined
and   ∃ instruction i which is not defined and which belongs to scenario
        element s
then  define instruction i'.
```

4. rules for defining relations

a) Defining relations, which are already part of the expert system database, is relatively simple: a relation between two scenario elements can be defined, as soon as both scenario elements are defined.

```
'if   ∃ relation r between scenario elements a and b
and   scenario element a is defined
and   scenario element b is defined
then  define relation r'.
```

b) It is also necessary to take indirect relations into account. For example, if the relations a-b and b-c exist (as part of the facts), but only scenario elements a and c are defined (so b is not included in the scenario), then the indirect relation a-c should be created and defined. For the creation of indirect relations, the following rule can be used:

```
'if   ∃ scenario element a which is defined
then  predecessors = find_predecessors_of (a) [3]
      successors = find_successors_of (a) [3]
      create relations between all scenario elements ∈ predecessors and a
      create relations between a and all scenario elements ∈ successors'.
```

Note that only relations, which don't yet exist, should be created.

c) By using a rule for defining indirect relations, it is possible that relations which have been defined before, become redundant. In the example above, the relation a-c becomes redundant, if scenario element b is defined (which causes relations a-b and b-c to be defined). The following rule can remove redundant relations:

---

2 Defining an instance of a class means, that this instance becomes part of the disaster recovery scenario.

3 'Find_predecessors_of' is a function, which searches all the defined scenario elements, which directly precede the given scenario element. To find these scenario elements the direct, undefined relations, which are part of the facts in the expert system database, are used. The function 'find_successors_of' searches all defined, succeeding scenario elements in a similar way.

```
‘if ∃ relation r1 between scenario elements a and b
and     ∃ relation r2 between scenario elements b and c
and     ∃ relation r3 between scenario elements a and c
and     scenario element b is defined
then    remove relation r3’.
```

<u>5. rules for defining relations that depend on the specific situation</u>

Some relations depend on the specific situation of the company, for which the disaster recovery scenario is set up (see section 4.1). These relations cannot be part of the facts in the expert system database, but should only be created if a certain condition is met. This can be accomplished by defining a rule for each conditional relation. To determine the value of a condition, a question can be asked to the user.

<u>6. rules for defining teams</u>

Some teams will be part of every disaster recovery scenario, while other teams will only be defined if necessary. The rules for defining teams will be similar to the rules for defining scenario elements.

<u>7. a rule for the assignment of scenario elements to teams</u>

For every scenario element, one or more teams can be specified to which the scenario element may be assigned. The assignment of the scenario element to one of those teams can take place, as soon as the scenario element has been defined. This requires only one rule:

```
‘if ∃ scenario element s which is defined and to which no team has been assigned
then    assign a team to s’.
```

<u>8. rules for specific computer systems</u>

As described in section 4.2, the expert system can be used to generate specific activities and detailed instructions for a couple of computer systems used often. These specific scenario elements and instructions have to be incorporated as facts into the expert system database. By the firing of rules, they can become part of the disaster recovery scenario, if it is known which computer system is used in a fall-back situation. For that, a question has to be asked to the user. (See rule types 1, 2 and 3.)

<u>9. a rule for the assignment of a location to resources</u>

The location of a resource, which is produced by an activity, is the same as the location, where the activity takes place. For such resources, the location can be set by the following rule:

```
‘if   ∃ resource r which is generated by scenario element s
and         ∃ scenario element s with location l
then        set the location of resource r to l’.
```

<u>10. logistic rules</u>

Logistic rules are related to the transport of resources (see section 4.3). They can be subdivided into rules for:

a) *generating transports*

Some activities require one or more resources for their completion. If the location of a resource differs from the location of the activity, then the resource has to be transported. The transport, as well as a relation between the transport and the activity which needs the resource, can be generated by the expert system. The following rule may be used [4]:

```
‘if   ∃ scenario element s which is defined, whose location is l1 and which needs
            resources rs
and         ∃ resource r which is defined and whose location is l2 ≠ l1
and         r ∈ rs
then        create a transport t from location l2 to location l1 for the
            transportation of resource r
            create a new relation between t and s’.
```

b) *combining transports*

If there exist several transports with the same start and end location, then they can be combined into one transport. (In case of such a combination, the corresponding relations should also be combined.)

c) *defining transports*

Transports which have been created, can be defined immediately. However, it is better to wait until transports have been combined. That is why the rule for defining transports should be given a lower priority than the rules for a) and b).

d) *relations concerning produced resources*

Some resources are produced by activities, which are part of the disaster recovery scenario. Those resources can only be transported, after the production activity has taken place. By means of a rule, relations can be created between production activities and transport activities.

4 This rule is only valid for non-exclusive resources (e.g. writing paper). For exclusive (unique) resources (e.g. a certain back-up tape), the situation is more complex, as such a resource cannot be transported twice from one location to another. We will not consider exclusive resources.

7. Discussion

To be able to decide if the expert system will be useful in practice, a prototype has been developed. For the development of the prototype, an expert system shell was used (see section 3.1). To get a clear picture of the usefulness of the expert system, we put as much knowledge as possible into the prototype. Little attention was paid to other parts of the system, like the user interface.

This resulted into a prototype, which realized most of the functionality mentioned in section 1. Part of the functionality was offered by the expert system shell and part of it was inherent to the chosen expert system technique (Rule-based Reasoning) and the included knowledge. Unfortunately, the shell offered only few possibilities for maintaining (adding, modifying, removing and reproducing) the knowledge. That is why we decided to store part of the knowledge in a separate file, which was loaded by the expert system during start-up.

Two experts on disaster recovery scenarios tested the prototype, to evaluate the usefulness of the expert system. Each of them chose a completed project in which he supervised the setting up of a disaster recovery scenario. By projecting himself again into that situation and by using his own knowledge on disaster recovery, he answered the questions, which were posed by the expert system. The scenarios generated by the (prototype) expert system, were compared with the actual scenarios which were set up in the projects concerned.

It appeared that the generated scenarios corresponded in broad outline with the actual scenarios. But as the generated scenarios can never be completely equal to the actual ones, further objective comparison was almost impossible. That is why the usefulness of the expert system was evaluated by interviewing the experts. During the interviews, a couple of observations came up:

1. The results of the answers, given by the user, are not clearly visualized. This problem can be solved by integrating the expert system with some kind of drawing software, which can graphically display a disaster recovery scenario.

2. It is not easy to correct a wrong answer. To correct a wrong answer it is necessary to undo all the results of rules that fired after the answer has been entered. Correction becomes easier if the state of the system is stored every time before asking a question.

3. The questions, posed by the expert system, force the user to think about many things, which are related to disaster recovery. This gives the user a good impression of everything, which should be described in a disaster recovery scenario.
4. Within a short period of time, a (draft) disaster recovery scenario is set up, which is rather complete.
5. The user is able to identify himself very well with the generated scenario. He has the feeling that it is *his* scenario. This is a significant advantage with respect to the adjustment of an impersonal generic scenario.
6. The expert system can also be used for demonstration purposes, to show how a disaster recovery scenario may be set up and what are the important issues.

8. Conclusion

In this article we described how to build a (prototype) expert system for setting up disaster recovery scenarios. In the previous section we discussed the results. From the results we conclude that such a system can be useful in practice. This conclusion refers to an expert system, which:

- offers the functionality mentioned in section 1;
- is based on Rule-based Reasoning (see section 3.2) and applies this technique as described in section 4;
- contains at least the knowledge described in section 6.

9. Acknowledgment

We would like to thank Mr. H.A. Overtoom and the consultants of the disaster recovery center CUC in Lelystad, The Netherlands, for providing us with knowledge about disaster recovery and for testing the prototype expert system in a real-life situation.

10. References

[1] Boullart, L., A. Krijgsman, R.A. Vingerhoeds, *Applications of Artificial Intelligence in Process Control*. Oxford: Pergamon Press, 1992.

[2] Vingerhoeds, R.A., B.D. Netten, D. Chitchian, A.J. Krijgsman, *Expert System Techniques - Rule-, Case- and Constraint-Based Reasoning*

(a223, supplement 2). Delft, 1995. Technical University Delft, Faculty of Technical Mathematics and Informatics.

[3] Bielawski, L., R. Lewand, *Intelligent Systems Design*. New York: John Wiley & Sons, 1991.

[4] Rumbaugh, J., et al, *Object-Oriented Modeling and Design*. New York: Prentice-Hall International, 1991.

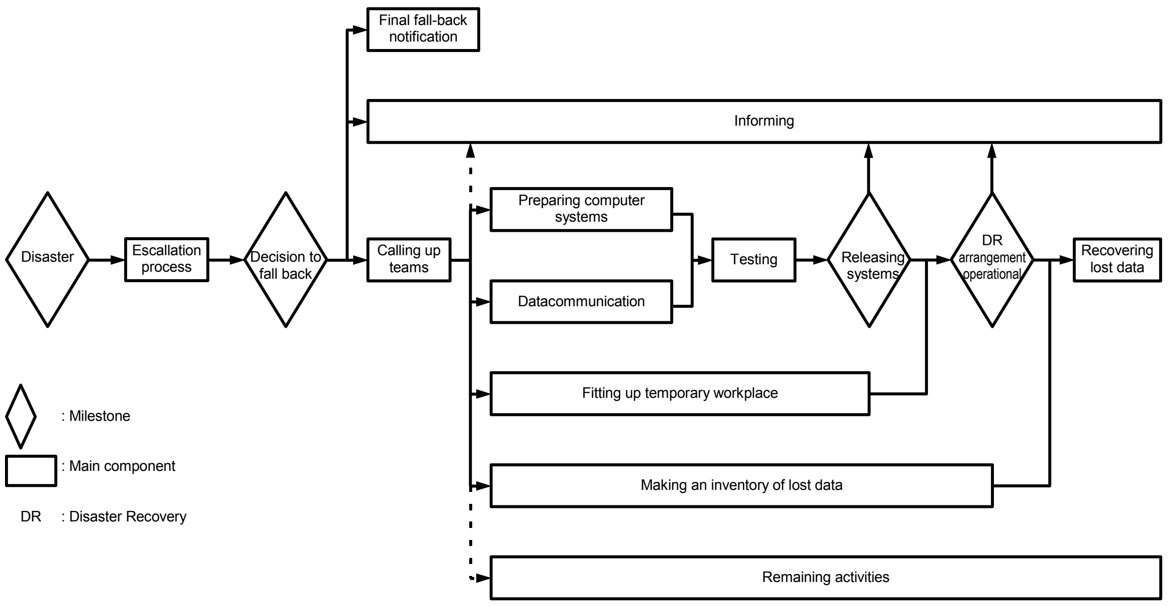

Figure 1: Framework of a disaster recovery scenario

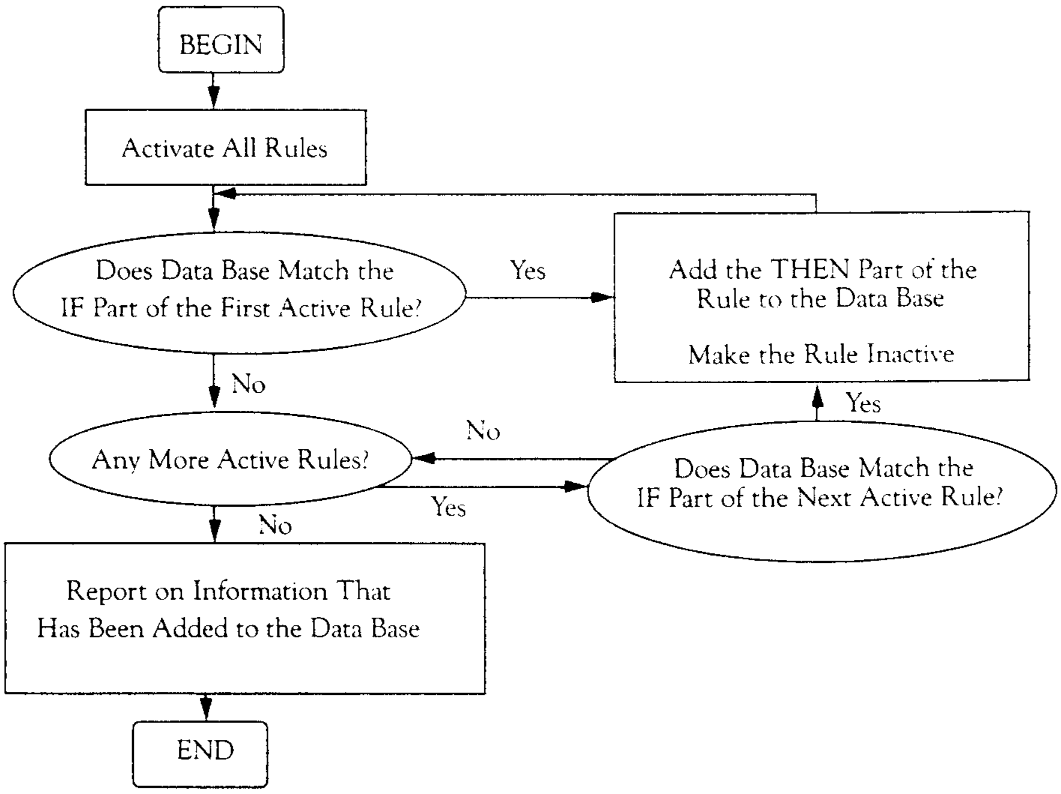
BEGIN
Activate All Rules
Does Data Base Match the IF Part of the First Active Rule?
Yes
Add the THEN Part of the Rule to the Data Base
Make the Rule Inactive
No
Yes
No
Any More Active Rules?
Does Data Base Match the IF Part of the Next Active Rule?
Yes
No
Report on Information That Has Been Added to the Data Base
END

Figure 2: Forward chaining

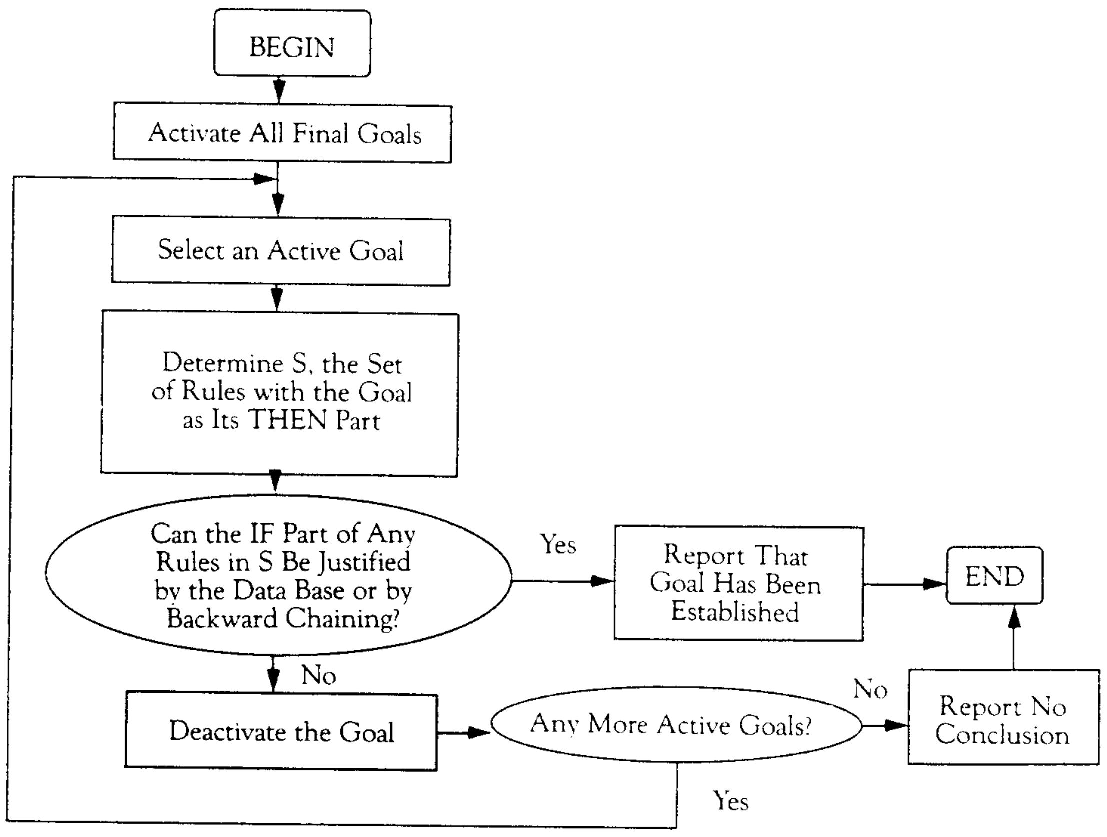
BEGIN
Activate All Final Goals
Select an Active Goal
Determine S, the Set of Rules with the Goal as Its THEN Part
Can the IF Part of Any Rules in S Be Justified by the Data Base or by Backward Chaining?
Yes
Report That Goal Has Been Established
END
No
Deactivate the Goal
Any More Active Goals?
No
Report No Conclusion
Yes

Figure 3: Backward chaining

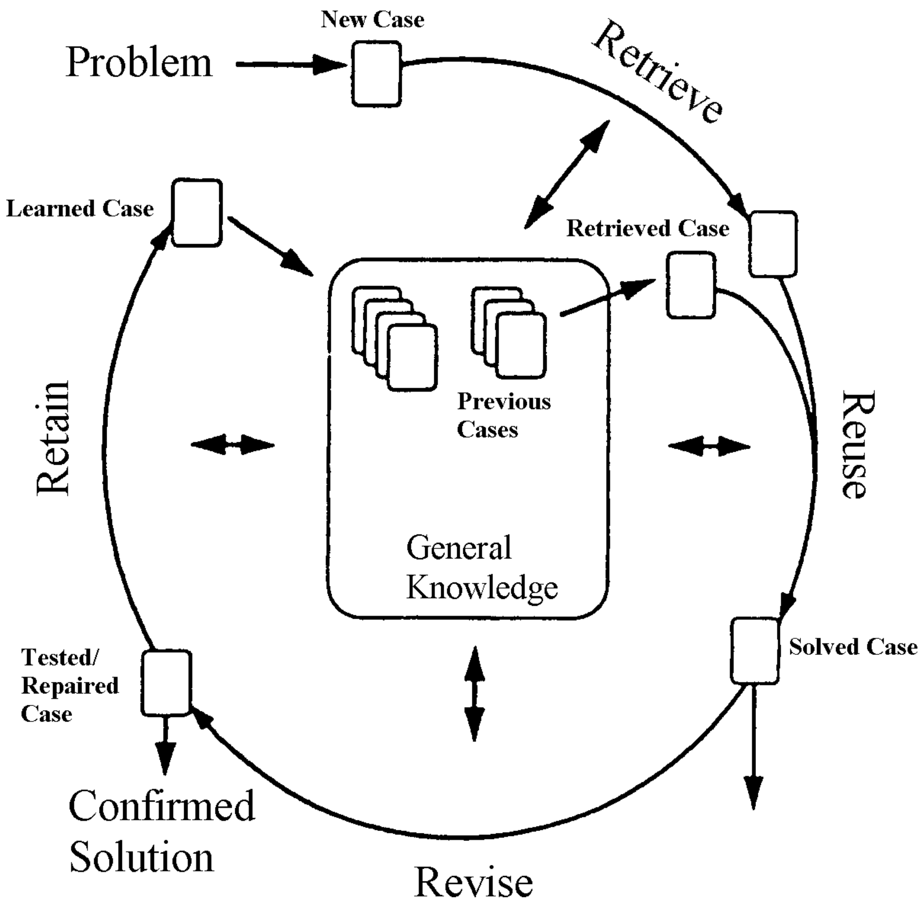

Figure 4: The Case-based Reasoning cycle

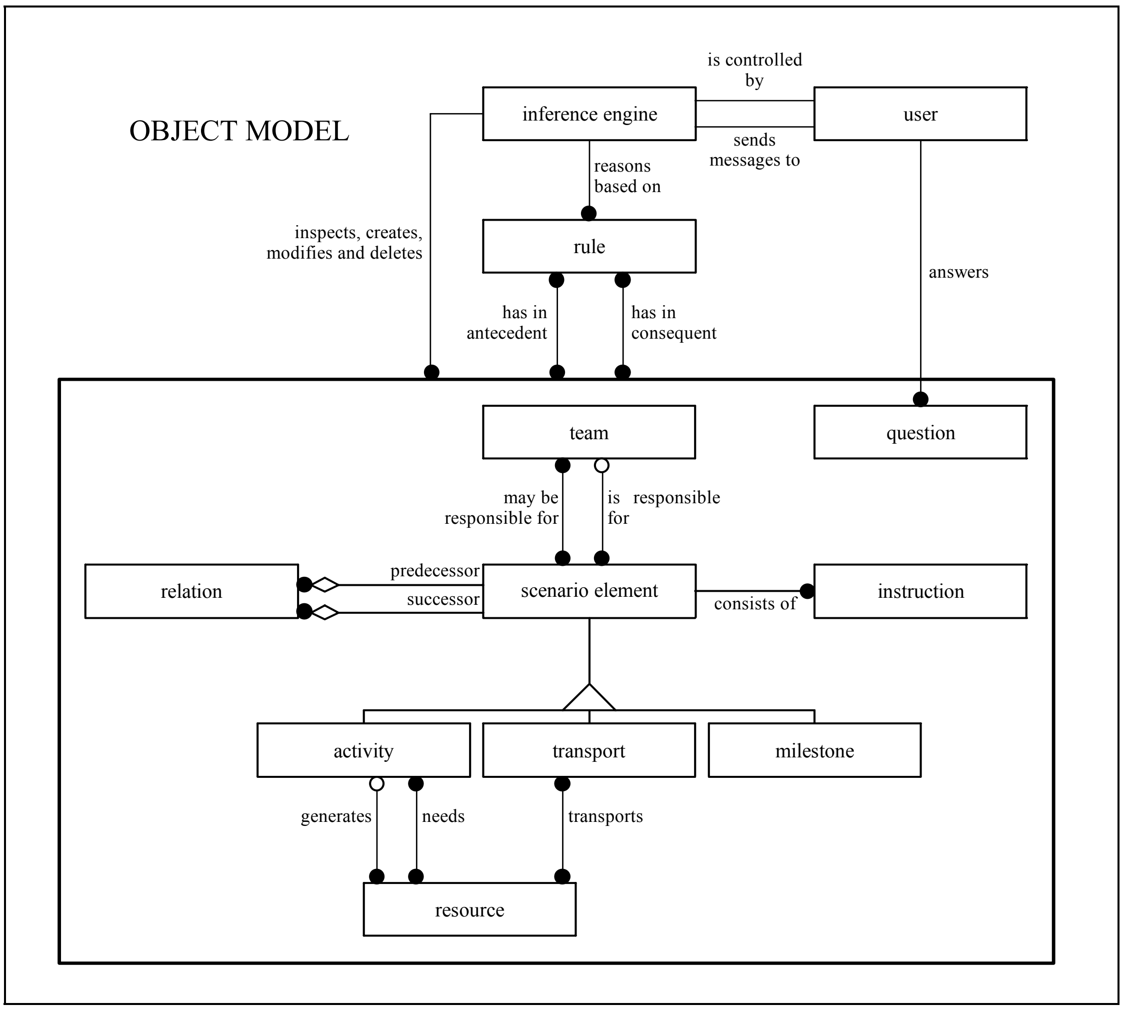

Figure 5: A sketch of the Object Model

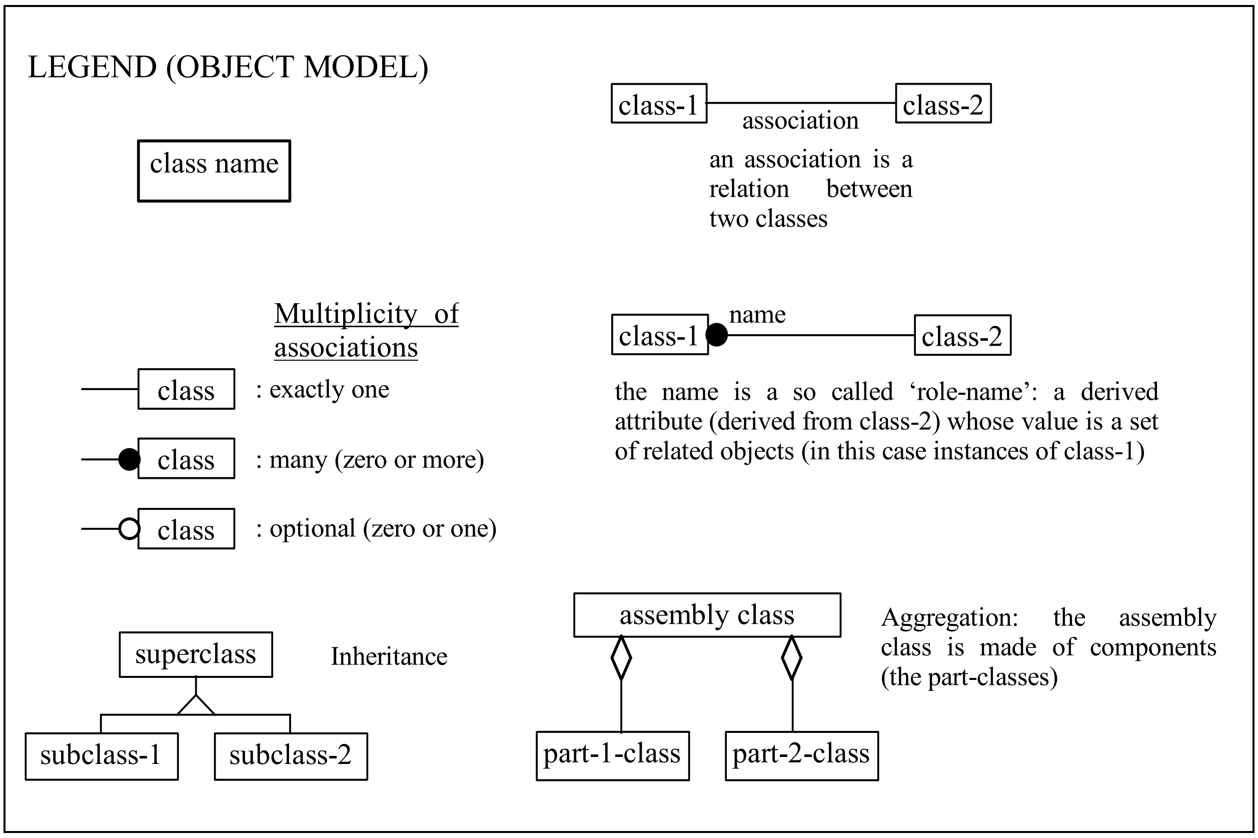
LEGEND (OBJECT MODEL)
class name
class-1
association
class-2
an association is a relation between two classes
Multiplicity of associations
class
: exactly one
class
: many (zero or more)
class
: optional (zero or one)
class-1
name
class-2
the name is a so called 'role-name': a derived attribute (derived from class-2) whose value is a set of related objects (in this case instances of class-1)
superclass
Inheritance
subclass-1
subclass-2
assembly class
Aggregation: the assembly class is made of components (the part-classes)
part-1-class
part-2-class

Figure 6: Legend of the Object Model