UPPSALA
UNIVERSITET

Department of Information Science
Division of Computer Science

Modelling with the User-Centred Knowledge Tool (t-UCK)

By

Anne Håkansson

2008

Modelling

Knowledge modelling is needed to structure domain knowledge before implementing the knowledge in knowledge base of a knowledge-based system (KBS). This modelling must be adapted to different kinds of users in general and in particular to knowledge engineers, domain experts and end users.

Many methods and techniques have been developed, but few of them are really effective and useful. Often the models containing the domain knowledge tend to be too large, making it is easy to loose the overview of the content. Moreover, the knowledge engineer has to be an expert in the modelling technique to be able to use the technique in the first place. Additionally, there are only a few modelling tools developed for knowledge-based system/ expert system and most of them are paper constructions and not automated.

These issues are the reasons for developing a new modelling tool, i.e., the User-Centred Knowledge tool (t-UCK). The model must be user-centred and support them in creating models of the domain knowledge. In t-UCK it is important that the knowledge is visualised. The tool visualises domain knowledge, reasoning strategies and functionality through graphic diagrams. To help remind the user of meaning and to make sure that the diagrams are useful, diagrams similar to the diagrams in UML (Unified Modelling Language) are used. UML is used for modelling object-oriented systems. The diagrams in t-UCK are modified to support modelling of production rules, which are used in rule-based systems.

Since t-UCK model is adapted for all kinds of users, the diagrams are not limited to use by the knowledge engineer to develop a system. Also the domain expert should use the diagrams to verify and validate the domain knowledge to assure correctness, consistency and completeness. The end users, on the other hand, should use the diagrams to understand the knowledge. Consequently, t-UCK is a tool used for modelling, implementing and testing (verification and validation), but also consulting (advisory) and education.

t-UCK model

t-UCK model has been developed over several years. The work with the modelling using UML began in 2000 and has been developed, changed and improved continuously. During this time several papers have been written and the implementation of several diagrams was done in Prolog and Java with Tcl/Tk.

The model has been used to develop several knowledge-based systems using rules, but also knowledge management systems (KMS). The technique for developing these systems is similar.

For more information, see papers and dissertation:

- Håkansson A., 2001. UML as an approach to Modelling Knowledge in Rule-based Systems. (ES2001) *The Twenty-first SGES International Conference on*

Knowledge Based Systems and Applied Artificial Intelligence. Peterhouse College, Cambridge, UK; December 10th-12th, 2001.

- Håkansson, A., 2003. Visual Conceptualisation for Knowledge Acquisition in Knowledge Based Systems. Accepted in: Frans Coenen (ed.): *Expert Update (SGAI) Specialist Group on Artificial Intelligence,* ISSN 1465-4091, 2003.
- Håkansson, A., 2003. Supporting Illustration and Modification of the Reasoning Strategy by Visualisation. (SCAI'03) *The Eighth Scandinavian Conference on Artificial Intelligence*, Bergen, Norway, November 2th-4th, 2003.
- Håkansson, A., 2003. *Graphic Representation and Visualisation as Modelling Support for the Knowledge Acquisition Process*. Ph D. Thesis Computer Sciences, Department of Information Science, University of Uppsala, Sweden. ISBN 91-506-1727-3.
- Håkansson, A., 2004. An Expert System for the Environmental Impact Assessment Method. Research report 2004:1. Universitetstryckeriet, Uppsala, ISSN 14-03-7572.
- Håkansson, A., 2005. Transferring problem solving strategies from expert to end users. (ICEIS-2005) *7th International Conference on Enterprise Information Systems*, Miami, USA, May 24th-28th, 2005.
- Håkansson, A., 2005. Modelling from Knowledge versus Modelling from Rules using UML. (KES-2005) *9th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems*, Melbourne, Australia, September 14th-16th

Outline

First a description of childhood diseases is presented. It contains the different diseases and the ones used in this material are measles, rubella, chicken pox, allergic purpura and cerebral membrane inflammation. The domain knowledge is obtained from a family lexicon for general diseases.

The next sections are images of the different diagrams that are used in the modelling. Each description begins with a general explanation about each diagram, clarifying the content in the diagram. Then an example of the diagram use is given in the domain childhood diseases. The diagrams presented are class diagrams with questions, conclusions and rules, object diagram, activity diagrams with sequence and collaboration diagrams. The diagrams do not cover all the rules for the childhood diseases because of the limits in space, readability and understandably.

The task used in this material is an example of a small knowledge-based system for childhood diseases. The system would handle diseases for children specialised on "Skin rash with fever". This means all different kinds of skin rash together with fever, which is 38 C or more.

The *questions* that would be asked are:

"Do you have large or small rash on the skin?" Yes / No
"Do you have two or more of these symptoms?" A runny nose / Bloodshot eyes / A dry cough
"Do you have any swelling on the back of the neck or on the necks side?" Yes / No
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" Yes / No
"Do you have only one rash that is a strong red spot?" Yes / No
"Do you have two or more symptoms? Vomit / Headache / Oversensitive to light / Pain when you bend your head forward

The *conclusions* are:

"You may have measles, see page 659, especially if the rashes is mainly in the face or on the trunk. See also the Coloured Picture 26, page 199."
"You may have rubella, page 660. See also Coloured Picture 27, page 199."
"This may indicate that it can be chicken pox, page 660. See also Coloured Picture 28, page 199."
"If you can not make any diagnosis by using this schemas, you should contact a doctor."
"Find a doctor immediately! The symptom can be a sign of allergic purpura, page 644, which is a blood disease."
"A VERY SERIOUS ILLNESS find a doctor immediately! The symptom can be a sign of cerebral membrane inflammation, page 388."

The *rules* given below are necessary if the connective "and", (conjunction), is used between questions or between rules together with questions. If the connective "or" is used, the number of rules can be decreased. The rules with using "and" are:

Rule 1:
Questions:
"Do you have large or small rash on the skin?" Yes
"Do you have two or more of these symptoms?" A runny nose / Bloodshot eyes / A dry cough
Conclusion:
"You may have measles, see page 659, especially if the rashes is mainly in the face or on the trunk.
See also the Coloured Picture 26, page 199."

Rule 2:
Questions:
"Do you have large or small rash on the skin?" Yes
"Do you have two or more of these symptoms?" No
"Do you have any swelling on the back of the neck or on the necks side?" Yes

Conclusion:
"You may have rubella, page 660. See also Coloured Picture 27, page 199."

Rule 3: (Could be a general rule to simply the knowledge base or written in all rules to as questions and answers.)
Questions:
"Do you have large or small rash on the skin?" Yes
"Do you have two or more of these symptoms?" No
"Do you have any swelling on the back of the neck or on the necks side?" No

Rule 4:
Questions:
"Do you have large or small rash on the skin?" No
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" Yes
Conclusion:
"This may indicate that it can be chicken pox, page 660. See also Coloured Picture 28, page 199."

Rule 5:
Questions:
The Rule 3 is correct and
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" Yes
Conclusion:
"This may indicate that it can be chicken pox, page 660. See also Coloured Picture 28, page 199."

Rule 6:
Questions:
"Do you have large or small rash on the skin?" No
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" No
"Do you have only one rash that is a strong red spot?" No
Conclusion:
"If you can not make any diagnosis by using this schemas, you should contact a doctor."

Rule 7:
Questions:
The Rule 3 is correct and
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" No
"Do you have only one rash that is a strong red spot?" No
Conclusion:
"If you can not make any diagnosis by using this schemas, you should contact a doctor."

Rule 8:
Questions:
"Do you have large or small rash on the skin?" No
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" No
"Do you have only one rash that is a strong red spot?" Yes
"Do you have two or more symptoms? Vomit / Headache / Oversensitive to light / Pain when you bend your head forward" Yes
Conclusion:
"A VERY SERIOUS ILLNESS find a doctor immediately! The symptom can be a sign of cerebral membrane inflammation, page 388."

Rule 9:
Questions:
The Rule 3 is correct and
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" No
"Do you have only one rash that is a strong red spot?" Yes
"Do you have two or more symptoms? Vomit / Headache / Oversensitive to light / Pain when you bend your head forward" Yes
Conclusion:
"A VERY SERIOUS ILLNESS find a doctor immediately! The symptom can be a sign of cerebral membrane inflammation, page 388."

Rule 10:
Questions:
"Do you have large or small rash on the skin?" No
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" No
"Do you have only one rash that is a strong red spot?" Yes
"Do you have two or more symptoms? Vomit / Headache / Oversensitive to light / Pain when you bend your head forward" No
Conclusion:
"Find a doctor immediately! The symptom can be a sign of allergic purpura, page 644, which is a blood disease."

Rule 11:
Questions:
The Rule 3 is correct and
"Do you have any red and itching rash, which is turning into blister that is full of fluid?" No
"Do you have only one rash that is a strong red spot?" Yes
"Do you have two or more symptoms? Vomit / Headache / Oversensitive to light / Pain when you bend your head forward" No
Conclusion:
"Find a doctor immediately! The symptom can be a sign of allergic purpura, page 644, which is a blood disease."

Class diagrams

Class diagrams are used as a pattern to assure that all the parts of questions, rules and conclusions are present. Those parts are the ones that, at least, are needed to handle domain knowledge in the knowledge-based system. Moreover, there has to be a relationship between questions, rules and conclusions.

Questions

”Class Question” contains all the parts that are required to ask a question. The signification parts are illustrated in Figure 1.

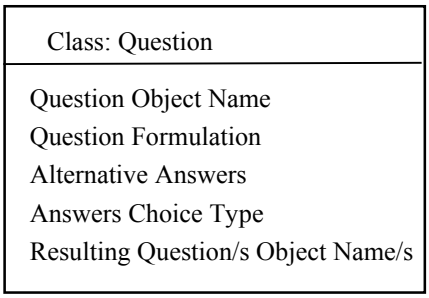

Figure 1. A general class for questions.

In the Figure, both the question’s head and content are illustrated. In the head ”Class: Question” is used to give it class membership. This is especially important when the domain knowledge is classified to show that some questions belong to certain rules.

In the body, several slots are specified. The ”Question Object Name” is used to relate a particular question to a rule. It is the object name that is specified in the rule when the rule is looking for an answer to a particular question.

The ”Question Formulation” is used to formulate the question posed to the users. The way the question is posed is important, i.e., it has to be clear and straightforward so the user can answer it properly without guessing the question and the answer to it.

”Alternative Answers” contains all the possible answer alternatives to the question.

”Answers Choice Type” shows what kind of answer alternative choice that is used in the question. It can be a single answer choice (s for single), multiple choices (m for multiple) or insert text or numbers (i for insert).

”Resulting question” implies that there is at least one question that is a following question, usually needed for asking for more information about the same topic. The following question is asked if the question’s answer alternative has a specific value. That means that if a question has a following question that is dependent on the first question, both have to be posed to the user. Observe that the following question has a corresponding slot – consequence question – that has a value that must be satisfied to be asked at all. Thus, the questions call each other but it is only one of the questions that must have a particular answer to be able to pose it. For more information, see below which is an example of a question and following question. Observe that there

are no requirements that the following question is specified in any rule. Instead, this following question should only be used when a specific question is asked. This applies for consequence question as well.

Objects of the question class

If the question class is a general template, it is possible to create concrete objects from the template. Hence, the template is used to fill it with information to suit the question that is going to be posed during consultation (executing of the system). The information is concrete for the specific question and situation and if there is some slot that does not suit the question, it can be omitted or left without data. Examples of questions are illustrated in Figure 2 and Figure 3.

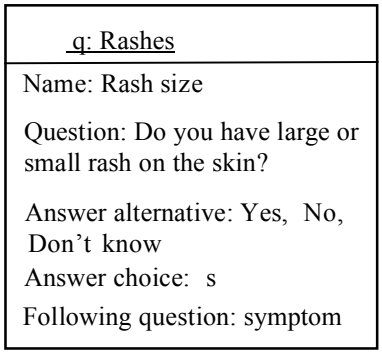

Figure 2. Question about rashes.

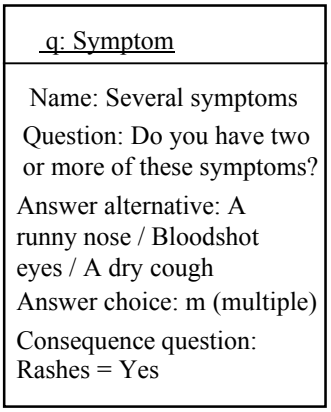

Figure 3. Question about symptoms.

In Figure 2, the class is denoted with a lower-case "q". The denomination is used to be able to put a relation between questions (q), rules (r) and conclusions (c) in an object diagram and rather easily distinguish the different object from each other. Moreover, the question in Figure 2 has the object name Rash size, which can be used to put several similar questions together and make a classification. This classification of questions and rules are a rather common occurrence.

The name on the question "Rash size" is specified in the name slot. This name is very important since it is used in the rules, so called fact objects. Thus, the question "Rash size" must have a specific answer to satisfy the rule with the fact object and, moreover, to be used in a conclusion, for example, the "Rash size" is "No". Instead for the answer alternative "Yes", "No", and "Don't know" the alternatives could have been "Small", "Large", and "Don't know".

The question's appearance is described in the second slot. It is exactly the appearance the question will have during consultation with the users. Observe, that it is not easy to ask a correct questions, so it is better to ask one question at the time. It is important to be concrete while determining the appearance of that question.

The possible answer alternatives are described in the third slot. It is very important to include an answer alternative like "Don't know". This alternative gives the user a way

to continue the session[1] if he or she has trouble with the question. If the user does not know whether there are any rashes or not, the user needs an answer alternative that covers this situation and the alternative "Don't know" covers many situations. The answer alternative is singular in this question, which means that the user can only answer with one alternative. In the last slot, the following question/questions are specified. If this slot is used, there is, at least, another question that is directly dependent of this question. In the example above, the "rashes question" has a following question "symptom" specified in the slot, which means that the symptom question will be asked later on in the consultation with the user.

All the names of the slots in questions, rules and conclusions can be omitted. However, if the diagrams are shown to domain expert or a user that is unfamiliar with the diagrams, it is good to let the name of the slots remain in the diagram. Those names can make it easier to understand the diagrams but also the domain knowledge in the system and how it is related to other parts of the system.

In Figure 3, the question about symptoms is presented. The name of the question is "several symptoms" and the question's appearance is specified in the second slot. Thereafter, there are three different answer alternatives with the answer choice m (multiple). In this case, the question can be answered with one, two or all the answer alternatives. In the last slot, the consequence question is specified which means that this question is only asked if the condition is fulfilled, i.e., the user answer "Yes" at the question about rashes.

[1] Session with the system, commonly, starts with a question and ends with a conclusion.

Rules

The production rules constitute the knowledge base, which corresponds to the domain knowledge. These rules specify the knowledge that can be used by the meta-interpreter during the execution of the system. Some of the rules are relatively detached and mainly make use of the facts produced by the questions. Some rules are strongly dependent of other rules, these rules together constitute a large network. This network can be complex in nature and difficult to comprehend. Figure 4 illustrates the general content of a rule.

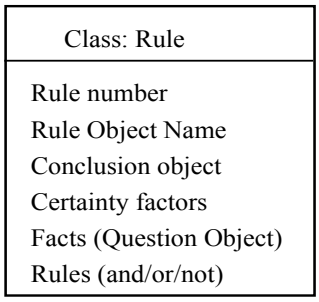

Figure 4. A general class for rules.

In the head of the rule, a name of a rule object is specified, see ”Class: Rule” in Figure 4. This name is used to classify rules when needed.

In the body of the rule, several slots are prearranged. In the first slot, number of the rule is given – ”Rule number”. This is the only data that makes the rule unique and this is needed when an individual rule is designated.
In ”Rule Object Name”, the name of the rules is specified. The name is used by other rules to refer to each other and also build a knowledge network. This network is interpreted in the same order as the rules are specified within each other.

”Conclusion object” is a slot that shows the conclusion that is reached when the rule is fired and been successful. All rules must have some kind of conclusion to be fired and has a conclusion object (or a kind of answer object) specified. One conclusion is usually used by several rules in the rule base and, in some cases, there is only one conclusion for the whole system, i.e., rule base

In the slot ”Certainty factors”, the value of the factor is specified. The interval for the certainty factors differs between the different data systems but it does not affect the outcome of the usage of the factors. Commonly, the intervals –1 to 1 and -1000 to 1000 are used.

In the slot ”Facts”, all the facts, used by the rule, is specified. These facts have to be satisfied to satisfy the rule. These facts are answers on the different questions that are significant for the rules. By using object name of the question and the answer alternative to that question, it is possible to connect the facts to the rules.

The ”Rules” slot is used to specify all the other rules that are connected with the rule, so called “the invoking rule”. The rules in the slot have to be satisfied to satisfy the invoking rule. The rules can either be specified with “and” which implies that these

rules have to be satisfied but the can also be specified with “or”. Or implies that the one of the rules in that set has to be satisfied but it does not matter which one. When using or, also several rules can be satisfied without making any difference in the interpretation. Also “not” can be used which means that the rule is not allowed to be satisfied. If the not rule is satisfied, the invoking rule fails.

Objects of the rule class

In the figures below, two different rule objects are illustrated. These objects are created from the general rule template and consist of more concrete data, se Figure 5 and Figure 6. The difference between these rules are that the rule in Figure 5 only use facts, while the rule in figure 6 uses other rules, in this case the rule specified in Figure 5.

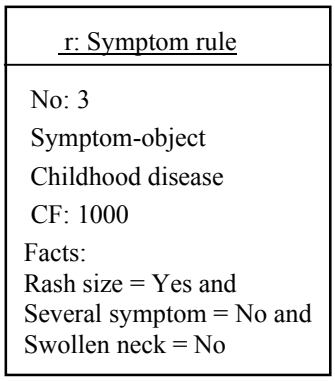

Figure 5. Rule for symptom.

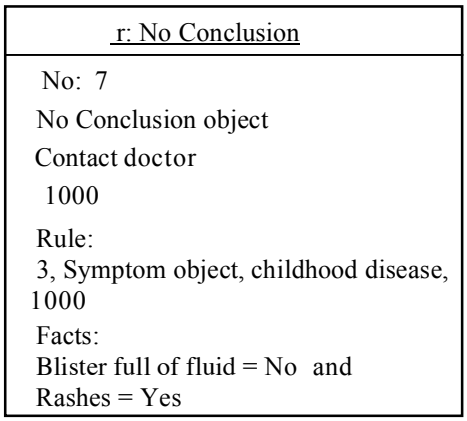

Figure 6. Rule for a case that is not deterministic for any diseases.

In figure 5, the rule for symptom is presented. In the head of the rule, is the lower-case ”r” used to denominate that the object is a rule. Also the name of the rule is given. In the first slot in the body of the rule, the rule number is given, i.e., No 3, but the name “No” can be omitted and, hence, only specify the number. In the second slot, the name of the rule object is given, which in this case is symptom object.

The conclusion, childhood disease, is specified in the third slot. This slot has at least one conclusion object (or answer object) that has the same name as the conclusion, in this case, childhood disease. The certainty factor (CF) has the value 1000, which implies that, it the rule is satisfied, the conclusion is correct with 100% certainty. This is only true for this rule. Even here is it possible to omit the name CF, like in the second rule object (Rule No 7), since it is only a clarification. Furthermore, some systems do not use certainty factors at all. In those cases the slot should be left empty.

In the fifth slot, the facts are presented, i.e., all the facts that the rule use. In this case it is the “Rash size” = ”Yes”, ”Several symptom” = ”No” and ”Swollen neck” = ”No”, see Figure 5 ”Symptom rule”. If these facts are stored during the session, the rule will be satisfied.

In Figure 6, a rule called ”No Conclusion”. It is a rule that handles the case where the facts or the rules do not satisfies the requirements for childhood diseases. Thus, it is not possible to decide whether the child has any childhood disease or not. The rule has number 7 as identification and the rule object is called “No Conclusion object”. If the rule is satisfied, the conclusion is “contact doctor”. Again the conclusion should be specified as a conclusion object (answer object) but there can be cases when a rule can only be used as a support rule for other rules. Then the rule cannot reach a conclusion by itself. It becomes a part of a set of rules that together reach a conclusion. The slot still needs to be filled to make the rule useful to others.

The certainty factor is 1000 and the facts in the rule are blisters and rashes. The user had answer “No” to “Blister full of fluid” and “Yes” to “Rashes”.

In the last slot, there is one rule specified, i.e., rule no 3. In Figure 6, the four first slots in rule no 3 are used to identify it in rule no 7. It is the rule number (3), object name (symptom object), conclusion (childhood diseases) and certainty factor (1000). It is information that the system uses during execution to interpret the rule base and reach conclusions. There are alternative solutions to using another rule, like rule 3. For example it is possible to declare the facts that the rule 3 contains directly in rule 7, i.e., “Blister full of fluid” is “No” and “Rashes” is “Yes”. However, only using facts instead of support rules is a poor solution and requires a lot of space in the rules. Moreover, when the number of rules grows, only using facts in the rules makes the rule base harder to change and maintain.

Conclusions

The conclusion that is presented to the users can be seen as answer or advice from the consultation. There can only be one conclusion per rule but the answer that is presented can differ depending on the consultation and the user-given answers. This is not the same as the system can only have one conclusion. On the contrary, the system can give many different conclusions by satisfying several rules.

Above reasoning is illustrated by following example. One main conclusion can be childhood disease but the conclusion drawn from the consultation can be measles, rubella, chicken pox, allergic purpura or cerebral membrane inflammation. The Figure 7 below illustrates the general template for the conclusions.

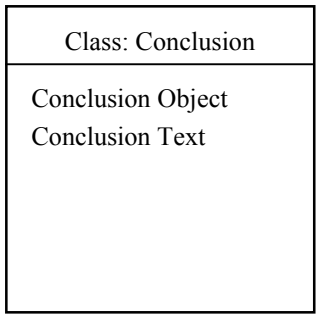

Figure 7. General template for conclusions.

In the head of the class Conclusion, the name for the conclusion is specified. This can be used to show who different conclusions are related, i.e., if the system can draw several different conclusions.

The conclusion object, see ”Conclusion object”, is used to connect the rules together with the conclusions. This means that the ”Conclusion object” corresponds to the slot three in the rules, see slot three in Figure 4.

The ”Conclusion Text” is the text that is presented for the user. The text describes the conclusion that has been reached.

Objects of the conclusion class

In the figures below, two objects of the conclusions are illustrated. These objects are created from the general template of conclusion, see Figure 8 and Figure 9. The lower-case c is denoting conclusion and the “Childhood disease” is the name of the conclusion.

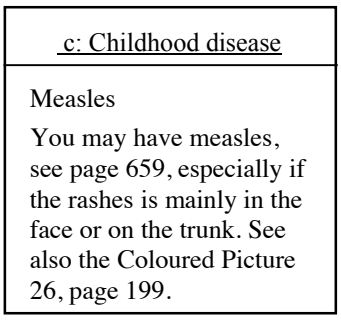

Figure 8. Conclusion for measles.

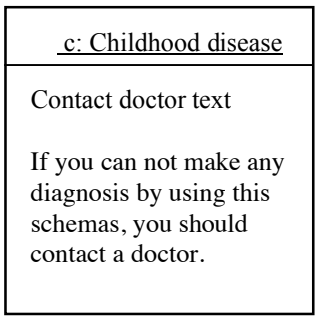

Figure 9. Conclusion for contact doctor.

The Figure 8 illustrates the conclusion for measles. When the system has reached this conclusion, the text: “You may have measles, see page 659, especially if the rashes is mainly in the face or on the trunk. See also the Coloured Picture 26, page 199.” is presented to the user. It is also possible to show the certainty factor within the text.

Figure 9 shows a conclusion that is presented when no childhood disease has been diagnosed. The text is: If you cannot make any diagnosis by using this schemas, you should contact a doctor.

If several slots are needed, is it possible to extend each template with the number of slots that are needed to cover the domain knowledge and represent the knowledge in the system.

Object diagram

The class diagrams are the building blocks for the parts that the system needs to function. However, an overview of the system is needed to see how the parts are related to each other. For this purpose, object diagrams are used.

The object diagrams contain all the objects that have been created from the template described above. To distinguish the objects, the denotations are used, i.e., q, r and c. Lines

are drawn between the different objects to illustrate which objects are dependent of each other. The objects together with the lines constitute a network of questions, rules and conclusions, see Figure 10. To make it clear how different parts in the diagram are connected, these are marked with bold font, see the heads of the objects in the figure.

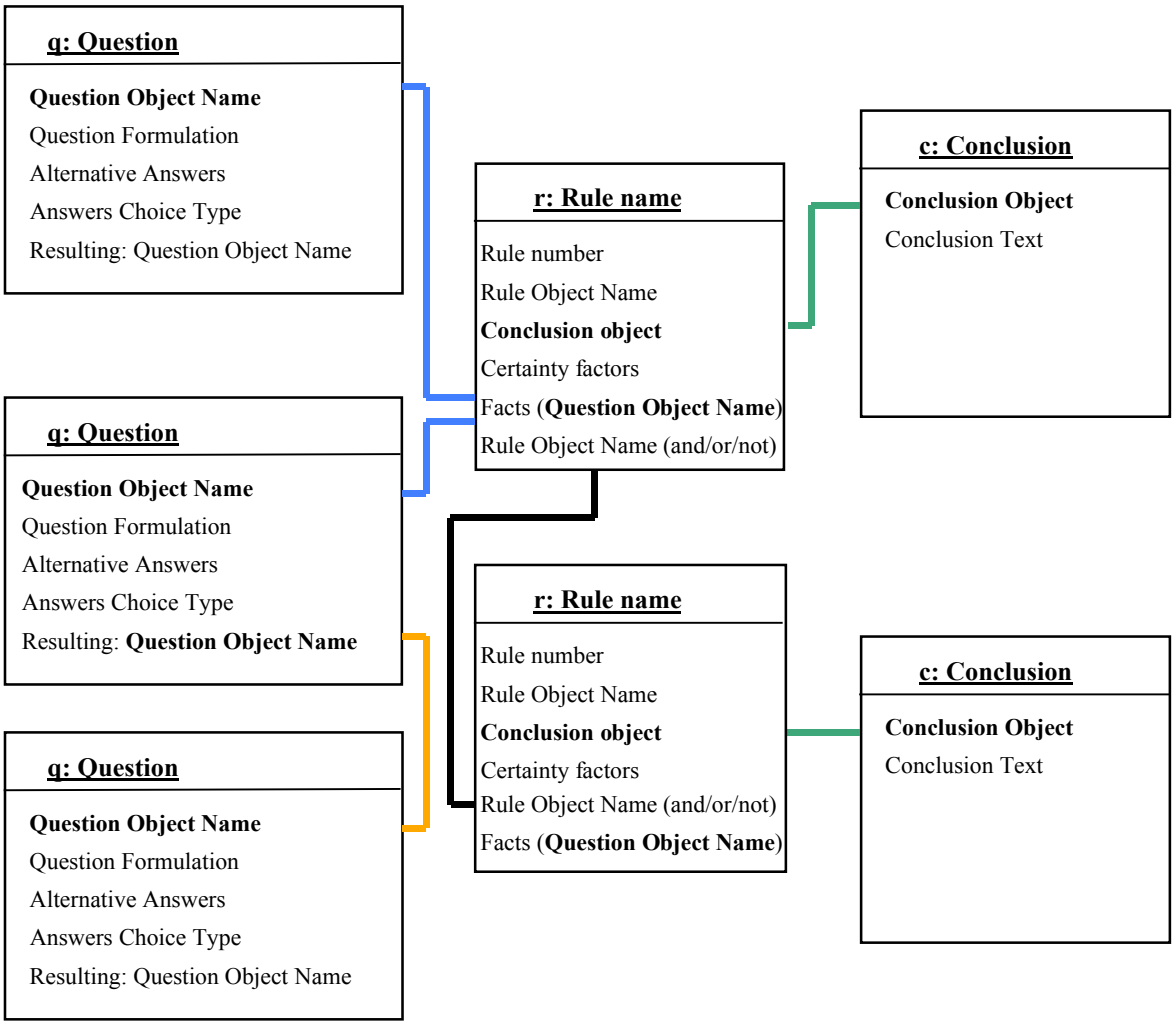

Figure 10. A general object diagram showing relationship between questions, rules and conclusions.

If the material is printed in black and white the lines between the objects are presented in different scales of grey. However, in reality, the lines have different colours. The reason for using different colours is to make it easier to see the connections between the questions (placed to the left), rules (in the middle) and conclusions (to the right). The lines between the objects also make it easier to discover if a part is detached without connections to other part. If so, the part is not used in the execution and do not fulfil any purpose in the system.

The line between a question and the following question (resulting question in the figure) is orange and the line between questions and rules is blue. The line between rules is black and the line between rules and conclusion is green. Using different colours on the lines are useful as long but in this case there is no guidelines for the choice of the colour. The most important part is to avoid red colour since they can be interpreted as stop or non

functional lines. Moreover, if green is used, red should be avoided since colour-blind people have trouble with and usually cannot distinguish between green and red colour.

In Figure 10, the most important parts are written in bold font. This is to elucidate the parts that are significant in order to make sure that the parts in the diagram are connected. For example, a question is connected to other questions and/or a rule/rules. It also shows how rules are connected to each other and how the conclusion is connected to rules.

For readability, the questions are always placed on the left side. In the figure there are three questions of which two also have to be posed to the user. The third on the other hand, placed down at the bottom in the figure, is dependent on the question in the middle. The rules are always placed in the middle. The rule placed at the top, uses both answers to the questions, the so called facts, and the rules that are specified under the facts. With a closer look at the rules, one can see that the slots for the facts and rules have changed place. The order of the slots should depend on how they are connected to other rules and questions. If there are lines that cross each other, the diagram can loose its readability since the diagram becomes complex and difficult to follow. Therefore, it is better to change the order of these and let rules sometimes be placed above the facts and vice versa.

Example of an object diagram

Since many parts are dependent on each other, the object diagram tends to become large, see Figure 11.

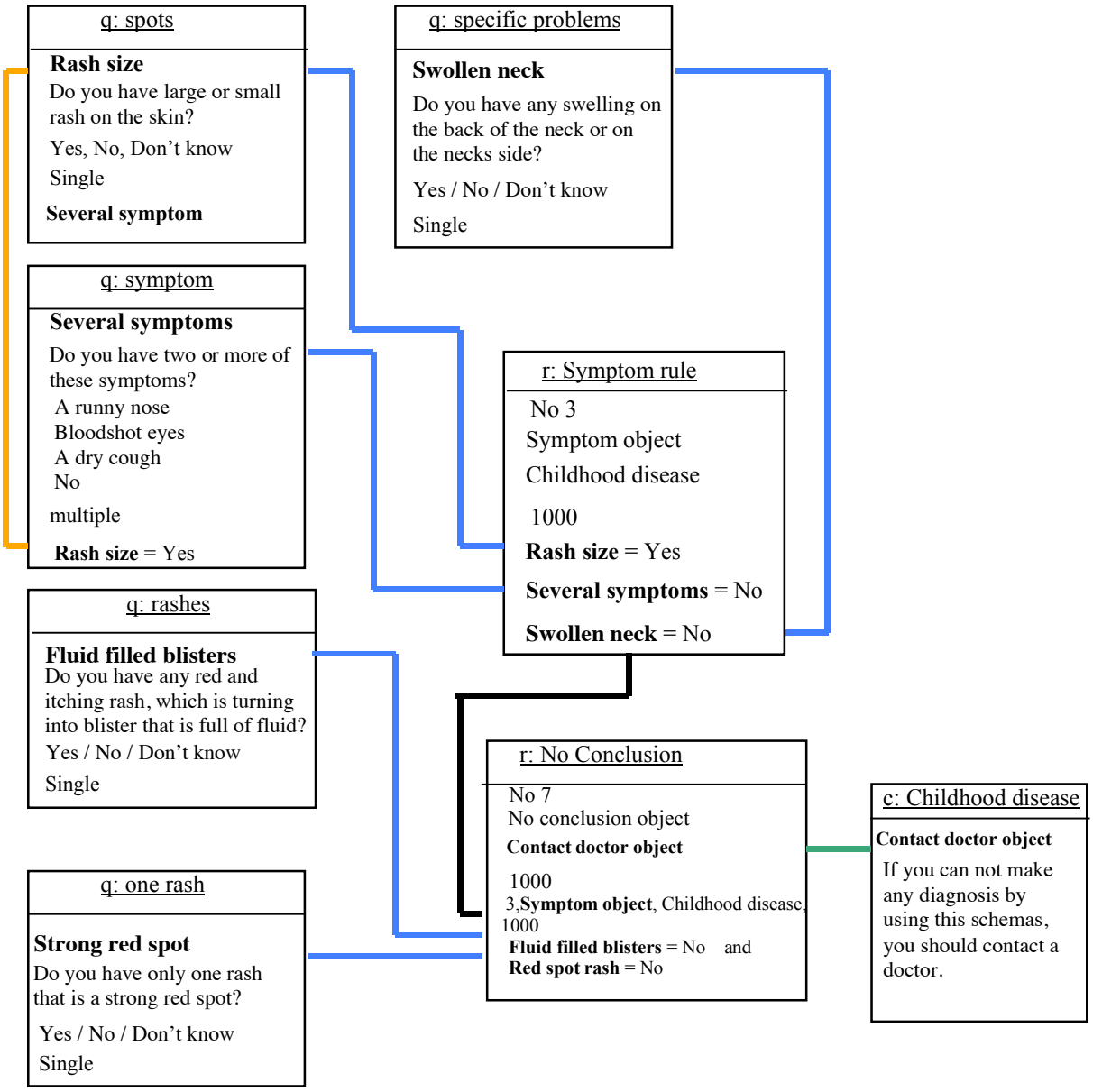

Figure 11. Example of an object diagram.

In the figure, five different questions are illustrated: spots, symptom, specific problems, rashes and one rash. To note that the objects are questions, q: is placed in front of the name. If this example was a large system, "rashes" and "one rashes" could be classified under the same name. Two rules are connected to the questions, rule no 3 "Symptom rule" and rule no 7 "No conclusion". Here "r:" is used to denote that the objects are rules. Finally, there is one conclusion "childhood disease". The conclusion object is denoted with a "c:" before the name.

Between the different parts, lines are used. As mentioned above, these lines have different colours and they are different depending the relationship. These colours are not determined but it is useful if the same colour is used between questions that are connected to other questions, questions to rules, rules to other rules, and rules connected to conclusions. The lines make it easer to grasp the overview, i.e., see the network of parts and how they are connected. Also it speeds up the process of grasping the content.

However, too much information makes it more difficult to grasp the content. Therefore, instead presenting all information in the picture, it is possible to use “packages” in UML. It is facility to embed information in units that is irrelevant for the moment but can be uncovered when needed.

To use packages, concepts are introduced. These concepts are used to describe questions, rules and conclusions with named or concepts. If the concepts are carefully chosen, they support the overview of the content of the object diagram. For simplicity the object names of the questions, rules and conclusions are used in the example, see the packages in Figure 12.

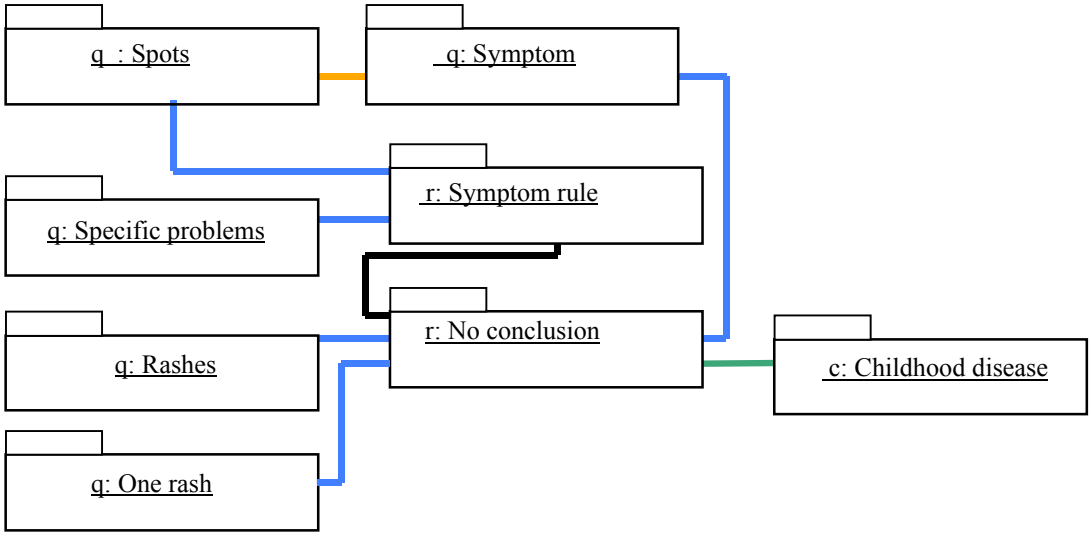

Figure 12. Object diagram with packages.

For the space in the diagram, the use of packages is effective. However, the packages can be difficult to understand if not all the parts are descriptive and obvious for those who have to interpret the diagram. This implies that the knowledge engineer can use these embedding facilities and still understand the system’s execution and purpose. The domain experts probably will have difficulties to grasp the overall picture and need the all parts presented concurrently.

Interaction diagram

There are two different interaction diagrams in UML, sequence diagram and collaboration diagram. Both these diagrams are used in t-UCK. The sequence diagram is used to show how rules are statically related to each other and the collaboration diagram is used to show how rules are dynamically related. Statically implies that the rules in the knowledge are always connected to each other in a certain fashion. Thereby, the sequence diagram is used to illustrate the network of rules with relationships. Dynamically means how the rules are fired during a consultation of the system. The firing of rules is affected by the answers given by the users. In some systems, these answers steer the communication with the system to some extent and, thereby, steering the firing order of the rules. The dynamic view of relationship differs considerably from the static view. There is a dependency between insertions of data in the dynamic connections of the relationship while the data insertions do not affect the firing or rules in the static relationship. The rules always have the static relationships regardless the user-given answers.

Sequence diagrams

The sequence diagram is used to illustrate rules in the knowledge base and the relationship between them. The diagram shows how the interaction is looks like for a use case, which in this case is the interpretation of the knowledge base. The interpretation of rules is in a specific order, which time wise shows how the interaction take place. Thus, the diagram can be used to illustrate how the rules invoke other rules or questions to finally reach a conclusion. A general example is illustrated in Figure 13.

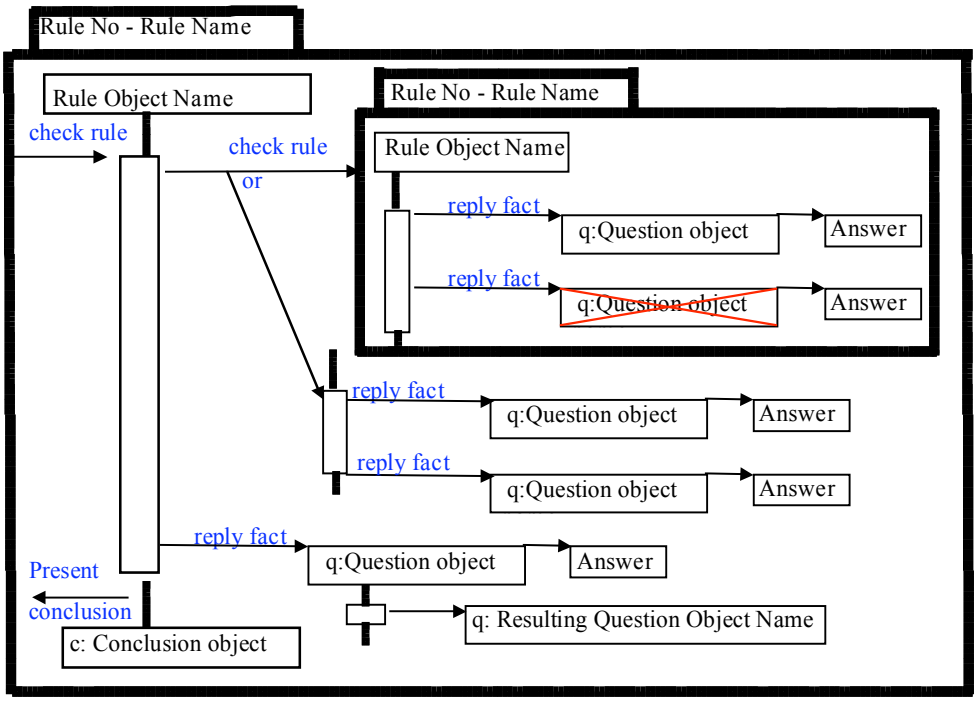

Figure 13. A General sequence diagram.

In the figure, the sequence diagram shows two rules that are dependent of each other. This is presented by the impressions of briefcases surrounded with bold lines. This supports using packages of rules, which is necessary when the knowledge base is expanding.

The overall rule is checked with invoking the rule with “check rule” in the beginning of the so call lifeline in the rule object ”Rule Object Name”. Everything in the lifeline should be interpreted as “and” between the objects and means that everything needs to be satisfied, that is, if nothing else is stated. The first step in that rule is, again, to invoke a rule with ”check rule. In this call there is an “or” which means that either the invoked rule or a question must be satisfied when executing the system. The line of the or-branch is split to indicate that there is a difference.

The rule has a rule number, “Rule No”, and a rule name, “Rule Name”, in the example above. The rule name is the same as the head of the object that is constructed from the rule class. In the package, the rule’s object name is given, “Rule Object Name”. From that another lifeline is present. Following this lifeline, two questions are invoked by using ”reply fact”. This means that when a question is invoked, the diagram controls that the fact corresponds to the answer that is required to satisfy this rule and continue to execute.

In this example, there is only one answer that is valid for this question, but it could be several answer alternatives (multiple choices) where, at least, one of these is correct.

The next call from the lifeline is also a question but in this case the box has a cross. This corresponds to "not" and means that the question must fail for the answer that is given in the box. Thus, the question is tested with a given answer but cannot be satisfied if the answer from the user corresponds to the answer in the box. If the question is not true, the rule is satisfied and the execution of the rule will continue. For example, if is said that: it cannot be true that the answer to the question "red rashes" is "yes". Then this does not mean that the question must be answered with "no". Rather it mans that the question can have any answer at all except for "yes". This correspond to "not red rashes"="yes". To control this condition, the question must be controlled by posing the question and receive the answer from the user. If the answer is correct (any this else than "yes") the rule is satisfied and can continue to execute. If the answer is "yes", the rule fails. In this case, failing rule means that the execution continues to the "or"-branch, which consists of two questions. Since these questions are at the same lifeline, both will be posed to the user.

When the rule or questions have been performed, the execution continues through the lifeline and there is yet another question that need to be asked before the lifeline is completed. Thereafter, the lifeline ends with a conclusion that is presented to the user. In the diagram, the conclusion is presented as conclusion object. This "conclusion object" must have the same name as the conclusion object created from the conclusion template, see "c: Childhood disease" in Figure 11.

Example of sequence diagram

In the example in Figure 14, a sequence diagram is used to show how rule number 7 uses rule number 3 and two different questions. The diagram contains exactly the same information as presented in the object diagram except for some details. It is only the names of the objects that are presented and not all the parts in the questions, rules or conclusions.

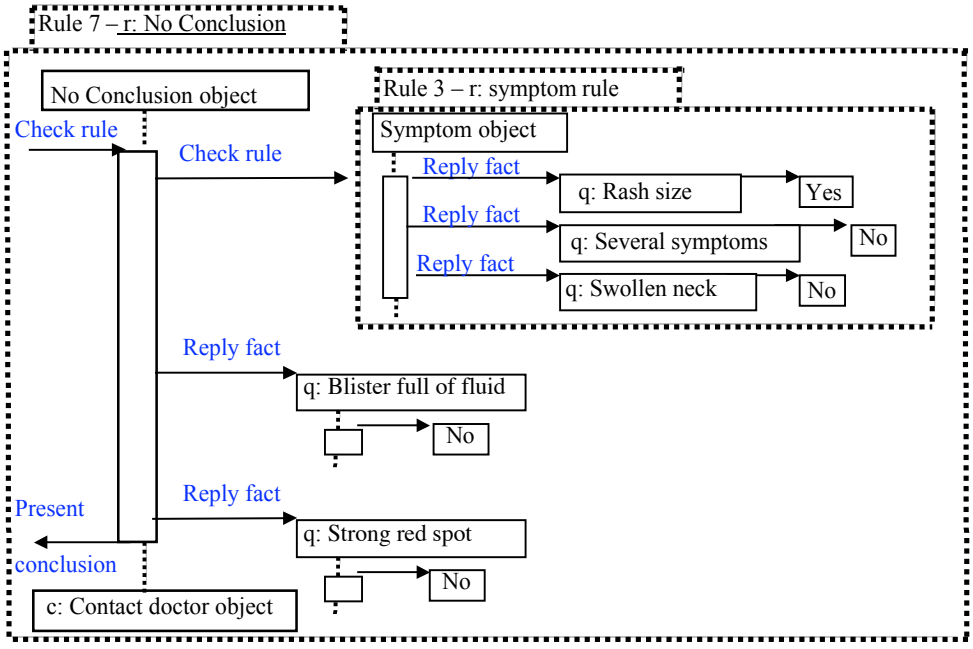

Figure 14. Example of sequence diagram.

Above the lifeline the rule 7 is presented with the rule name r: and "no conclusions". This diagram starts with checking a rule, which has three facts that have to be satisfied (three questions with answer alternatives that correspond to the facts in the rule). In this case, the "Rash size" has to be "Yes", "Several symptoms" has to be "No" and "Swollen neck" has to be "No". After the rule has been checked, the question of "fluid filled blisters" is checked to control if the answer is "No" and also the questions about "Strong red spot", which also must be "No". When the rules and facts connected to the lifeline are examined, the conclusion is presented to the user. In this case, it is the text attached in the conclusion object "Contact doctor text". This is illustrated as "Present conclusion" and an arrow in the end of the lifeline.

Again packages can be used to support comprehensibility. Usually this facility is only needed for rules because questions and conclusions are seldom complex.

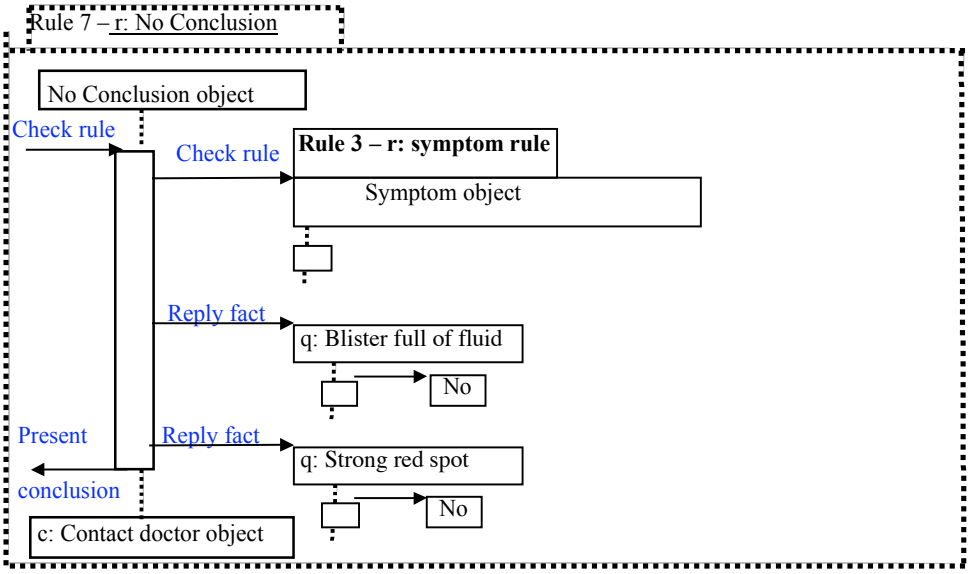

Figure 15. Example of packages in sequence diagram.

In the example above, one rule has be packaged and named "Rule 3 – r: symptom rule". The same name is used to support remember the content and purpose of the rule in the first place.

Collaboration diagram

The collaboration diagram shows how different objects collaborated with each other during execution of the system. In a collaboration diagram, it is possible to illustrate which interactions, between objects, take place for the use cases, but the diagram does not have any time order as in sequence diagrams. However, it is possible to show the order between objects from calls to the diagram. Usually, the collaboration diagrams have a static part and a dynamic part. The static part describes the acts of the objects while the dynamic part contains of interactions that shows how messages are passed between the different objects.

In this modelling tool, the focus is on the dynamic part. The collaboration diagram is dynamic in the sense that it changes for each consultation as long as the user does not give

exactly the same answers to the questions. The diagram adjusts from the call and illustrates the parts that are involved in that call. By this, it can be easier to get an overview of how different premises contribute to a conclusion. Visualising this process support grasping the execution order of rules and facts and, thereby, also the reasoning strategy of the system.

Note that it would be nice to reflect the domain expert's reasoning strategy within these diagrams but, commonly, the reasoning strategy of the system and of the domain expert differ due to technical limitations.

A general collaboration diagram has following layout, see Figure 16.

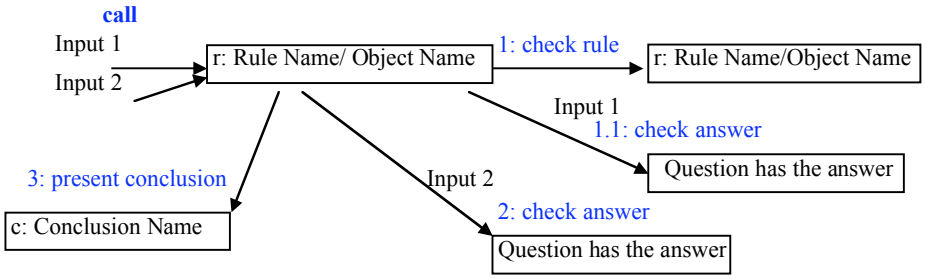

Figure 16. General collaboration diagram.

Input to the diagram is the answers to different questions and this is denoted as a "call" to the diagram. From the answers, it is possible to check the rules that have parts corresponding to the input. It is also possible to control if there are questions that have answers that correspond to the input (facts).

In Figure 16, two different inputs are given, that is, value or answer of two questions " Input 1" and "Input 2". From these inputs, a rule is invoked, "r: Rule Name/ Object Name" in upper-left in the figure. Since this rule uses another rule, there is a new call with "1: check rule" to "r: Rule Name/ Object Name" to the right in the figure. From this, it is a call to a question "Input 1" that is within the rule why it starts with the same number "1.1: check answer". Next call, "Input 2", is to a question external to the rule, why it starts with new number "2: check answer". When all the parts are inspected, the conclusion becomes the output from the diagram, i.e., "3: present conclusion".

Example of a collaboration diagram

The example that is illustrated in the Figure 17 is based on the same material as in the sequence diagram.

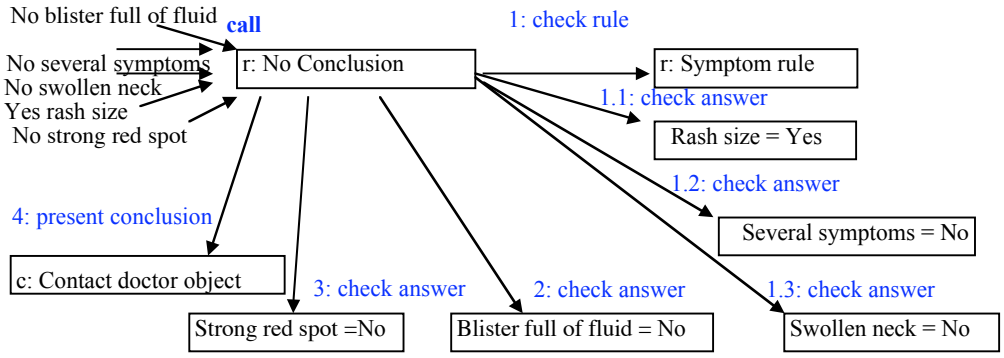

Figure 17. Example of a collaboration diagram.

In the upper-left corner in the diagram, the calls to the rule "No conclusion" are specified. The calls are "Blister full of fluid" with the answer " No", "Several symptoms" is "No", "Swollen neck" is "No", "Rash size" is "Yes" and "Strong red spot" is "No". These facts (answers) are used when the rules and the questions are tested in the diagram.

First the rule "r: No conclusion" is invoked. From this rule, there are several calls. Among these calls, there is another rule with its facts. The diagram tests the other rule "r: symptom rule" together with three facts. It is "Rash size = yes", "Several symptoms = No", and "swollen neck = No". Since all these facts belong to the same rule, the number starts with the same number as the rule. The rule has the number 1: and the facts 1.1:, 1.2: and 1.3:. When the rule is accomplished, the testing proceed to the to questions, 2:check answer, where the "blister full of fluid = No" and 3: check answer where the "strong red spot = No". When this is finished, the conclusion "c: Contact doctor object" is presented to the user.