NTNU
Innovation and Creativity

Conversational CBR for Improved Patient Information Acquisition

Tor Henrik Aasness Marthinsen

Master of Science in Computer Science
Submission date: June 2007
Supervisor: Agnar Aamodt, IDI
Co-supervisor: Arild Faxvaag, St. Olavs Hospital

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

In this thesis project a CCBR system for gathering of patient information shall be designed, and partly implemented. The focus will be on developing a method for generating questions targeted to a particular patient's reaction on medication, based on other patients similar experiences.

Essential parts of the architecture shall be exemplified in a demo. In a recent PhD research project methods for CCBR were developed that should be reviewed and analysed, and used as input to the methods developed in this thesis. The implemented system will be based on an existing core CCBR system, implememted in C++. This thesis project is linked to an ongoing cooperation between IDI's KBS group and NSEP, with the goal of exploring the use of CBR for electronic health records.

Assignment given: 18. January 2007
Supervisor: Agnar Aamodt, IDI

Abstract

In this thesis we describe our study of two knowledge intensive Conversational Case-Based Reasoning (CCBR) systems and their methods. We look in particular at the way they have solved inferencing and question ranking. Then we continue with a description of our own design for a CCBR system, that will help patients share their experiences of side effects with drugs, with other patients. We describe how we create cases, how our question selection methods work and present an example of how the domain model will look. It is also included a simulation of how a dialogue would be for a patient.

The design we have created is a good basis for implementing a knowledge intensive CCBR system. The system should work better than a normal CCBR system, because of the inferencing and question ranking methods, which should lessen the cognitive load on the user and require fewer questions answered, to reach a good solution.

Acknowledgements

I would like to thank my supervisor Agnar Aamodt for his advice, discussions and help during the work with this thesis. Also I would like to thank Arild Faxvaag for his advice and input on the medical parts of the system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis studies the possibility of using Conversational Case-Based Reasoning (CCBR) in a system to help share patient experiences about the side effects of drugs. This will be done by looking at related research within this area and studying CCBR methodology to find the best solutions for the system.

1.1 Motivation

Patients often want to know how other patients have responded to the treatment they are getting. We want to create a system where patients can share their experience with a drug with other patients. With the use of the Internet for gathering information, also about medical issues, rapidly growing among the average population, this system can give the patients the possibility to check if side effects of the drug they are taking can be responsible for a new illness they are experiencing and possibly give a solution that has worked for similar patients.

Case-Based Reasoning (CBR) is an excellent method to use for reasoning on this kind of data. Each treatment of a patient would be a case and it would have features such as what drug the patient is using, any additional medicaments the patient is using, any side effects experienced and so on. These features can then be weighted and you can then use these cases to reason what a good solution could be to a new case.

One problem with using ordinary CBR is that the users, here the patients,

would have to know how to fill out a complete and correct case to get a good solution. Since the patients are not experts at this system and should not have to be, they need to get help with this. A solution to this problem is using CCBR. Here you only start with a partial case and from this case the system asks the user questions to help fill out a correct query. This means that you will have an interactive process until the system provides a case solution that is acceptable by the user, or the system has no more questions to ask to make a better solution than the cases it is already presenting.

With Mingyang Gu finishing his PhD within the CCBR field there has been an increasing interest for the subject in the group for intelligent systems here at NTNU. There are 4 students working on master thesis within the field this spring and in cooperation with Trollhetta AS it has been decided to try to develop a new CCBR system from scratch. The other three master students are working as part of a large European project for Improved treatment of pain, depression and fatigue through translational research under The European Palliative Care Research Collaborative.

1.2 Goal

The goal of this thesis is to look at the requirements and the possibility of making a CCBR system which allows patients to share their experiences of side effects with using pain-relievers. This includes:

- Designing case structures
- Construct example cases
- Designing methods for inferencing and question ranking
- Describe how the domain model should be
- Test cases in the new CCBR core from Trollhetta AS

There will be a need to study different articles about CCBR to determine what kind of functionality is needed to create a good system and which solutions are preferred. It will also be necessary to look at existing medical CBR and CCBR systems.

1.3 Medical view

It is essentially three questions we want the system to be able to give an answer on, to a patient. The first is if the problem the patient is experiencing could be caused by side effects of a drug the patient is already using. This will have to be based on what is known side effects of the drug. The second question is if the patient should cease the therapy with the current drug or not. In some cases the best alternative is to continue with the drug even though the patient experiences side effects. This will of course depend on what the side effects are, what the treatment is for and if there are any alternatives to the drug the patient is currently taking. The last question is if the patient decides to cease the treatment with the current drug, should he or she change to a different drug or end the medical treatment.

A lot of the solutions that will be presented to the patient, will require the patient to consult with a doctor before making a drug related decision. For all drugs that are subject to a prescription, any changes to the medical treatment will have to go through their physician. The patient will however be able to see what has worked for other patients with similar cases and be better informed about his own illness when discussing with his physician, what to do next. For non prescription drugs, the solution offered may be to switch to an alternate drug.

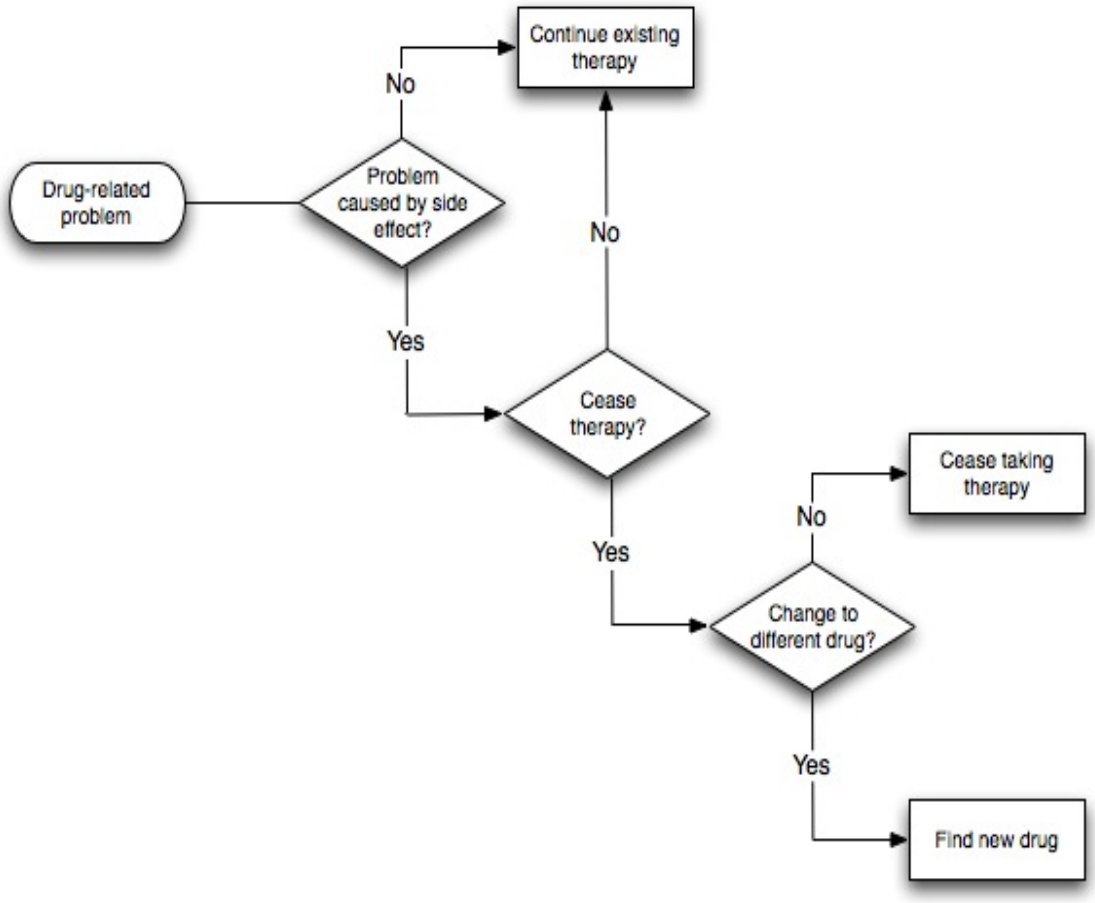

Figure 1.1: Medical view

1.4 Research method

This thesis will be based on a study of CCBR systems, in particular the one described by Gu and Aamodt in [13], and their methods. The focus will be on creating and describing the design of a CCBR system for patients to share their experience of side effects. This will be done with a basis in the drug information found in Felleskatalogen [1]. We will present a simulation of how an experience with the system would be. The cases will be implemented in the C++ core, from Trollhetta AS, to test if they are compatible.

1.5 Overview

The rest of this thesis is organized as follows. In chapter 2 we present the background CBR and CCBR theory, with emphasis on inferencing and question ranking. Then in chapter 3 we describe the system design and in chapter 4 we show an example of a simulation of a dialogue with a user. In chapter 5 we have final discussions, related research and our conclusions.

Chapter 2

Theory

2.1 CBR

Case-Based Reasoning (CBR) is a method for solving problems, were you use similar experiences to derive a solution for a problem you are facing [3]. These older experiences are stored in cases. The system then checks for similarities between old cases in the casebase and the new problem case, and retrieves the best matches. This case matching can be done by a simple comparison of each feature in the query, with each feature in the different cases in the casebase. A more advanced solution is using knowledge intensive reasoning by adding a domain model with explicit general domain knowledge [2].Then this best match case or cases are reused to form a solution to the problem at hand. This solution is then revised to adjust it to the current problem. And finally the system can retain the new solution case for later use, by storing it in the case base. This forms the well known CBR cycle, seen in figure 2.1.

Each case is comprised of several features describing a problem and a solution, if it is a solved case. These features each have a value and together they make a feature value pair. If all the values are either empty or have one discrete value, this can be numeric or a string, the case is called a point case. For several systems you may want to have intervals as a feature value. To be able to handle this you need to use generalized cases, were each feature can have multiple values [7].

The domain model is a way to enter known general knowledge from the domain experts into the system, so that the system can use this knowledge

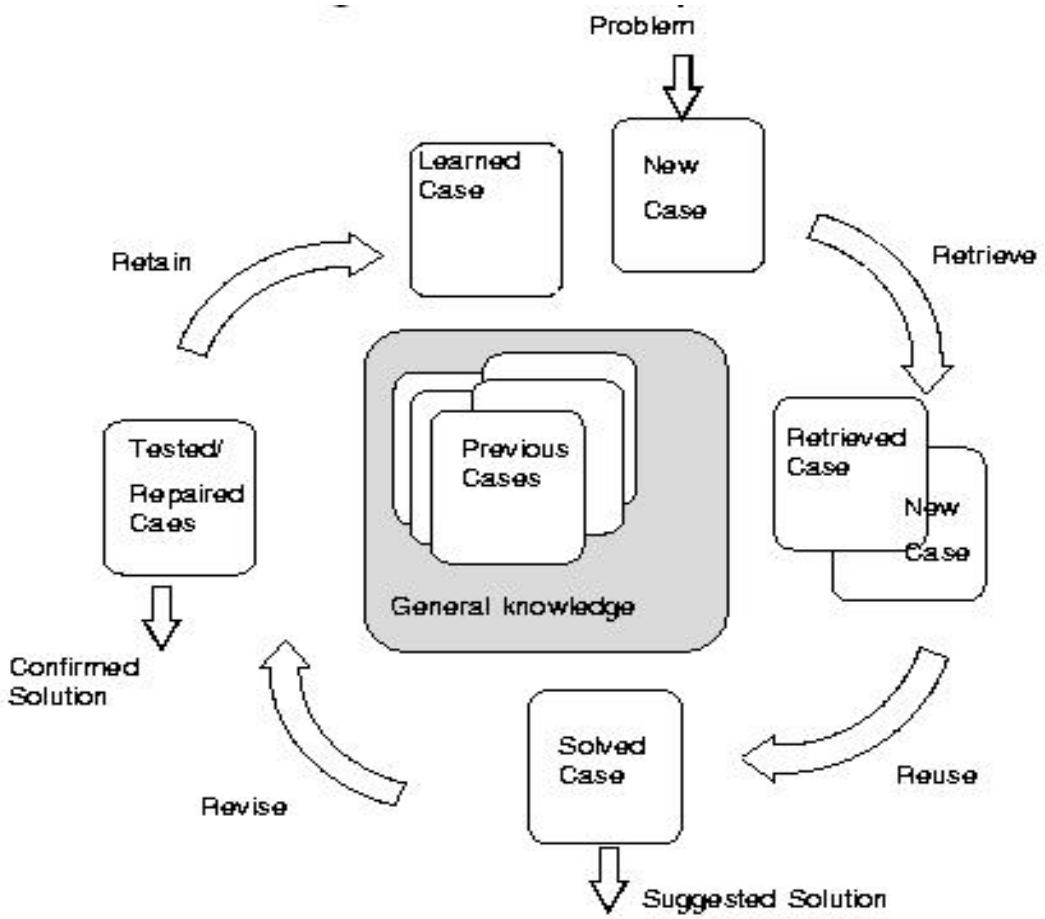

Figure 2.1: The CBR cycle

to present a better solution to a proposed question. This can be done by filling in this knowledge in a set of entities and relations, with the relations binding the entities together. With the different relations you have different qualities shared between the entities, such as inheritance or causality.

2.2 Conversational CBR

CCBR is a version of CBR in which the user participates in an interactive dialog with the system to create the query used in case-matching. The user therefore does not have to know which features are important for the system

or why they are important. The system will supply a list of questions, from which the user will be able to select which ones to answer. This have made CCBR systems work very well in real world implementations, where the users are not necessarily domain experts and there are a lot of uncertainties that are hard to predict while developing the system [4].

CCBR was first used by the Inference Corporation in their product line for customer support tools [6]. There have since been several other systems like the Casebank Technologies use of CCBR in diagnosis of jet engines. In the recent years there has also been a large increase in research programs in this field. Aha and Gupta looked at the possibility of using taxonomies to help with different abstraction levels, and causal relations between the taxonomies to better performance even more [14] [5]. Gu and Aamodt looks at more relations among concepts in [9] and the possibility of generalized cases in CCBR in [11].

As with a normal CBR system, a CCBR system can learn from storing new solved cases in the casebase, which will enable these solutions to be used in finding a solution to any new problems entered. With CCBR there can also be some value in storing the question/answer path that was selected. This can then be used to rank questions chosen by previous user in the same situation higher on the questions list presented to a user on a new problem [12].

This advantage in user-friendliness also adds a few design problems. Like how the system decides which questions to present to the user? The system should ask the user as few questions as possible to arrive at the best solution as fast as possible.

2.2.1 Question selection in CCBR

In CCBR an important part of the system is which questions to ask and in what order to ask them. The user should experience that the questions are coming in a natural order and should not be asked unnecessary questions. To get the best possible case match, as quickly as possible, the CCBR system should also select the most discriminative questions.

Semantic relations

In [9] Gu and Aamodt look at how this problem can be solved in a knowledge intensive system. They identify two ways to do this, where one is to remove questions where the answer can be inferred from information already provided by the user. The second way is to rank the remaining questions and there are several criteria to use for this purpose. In the general domain model they identify five semantic relations that are used to infer new information from the information provided, help rank questions or both.

The first relation is concept abstraction which makes a hierarchy with subclasses, as in Volvo is a subclass of car. If we have A is a subclass of B, concept abstraction can be used to infer B if we already have A which means that we don't have to ask about B. If we don't have either A or B concept abstraction can used in question ranking to decide that B should be asked before A, because if we don't have B we will not have A.

Dependency relations defines that one concept depends on the existence of another concept. If we have A depends on B, this can be used in question selection to infer B if we already have A, or decide that B should be asked before A if we don't have either A or B.

Causality relations defines that one concept causes the occurrence of another concept. If we have A causes B, then this can be used in question selection to infer B if we already have A, or that A should be asked before B if we don't have either A or B.

Correlation relations are used to define two concepts that always occur together. If you have A correlates with B, then you can infer A if you have B or you can infer B if you have A. This can not be used in question ranking.

At last we have Practical costs which represents a way to say that one piece of information is more costly to figure out than another. This can be used in question ranking if we have A is more costly than B then B should be asked before A.

These five semantic relations are placed as subclasses under infer and follows in the semantic relation hierarchy, as seen in figure 2.2. The infer relation is used for dialogue inferencing and the follows relation is used in question ranking.

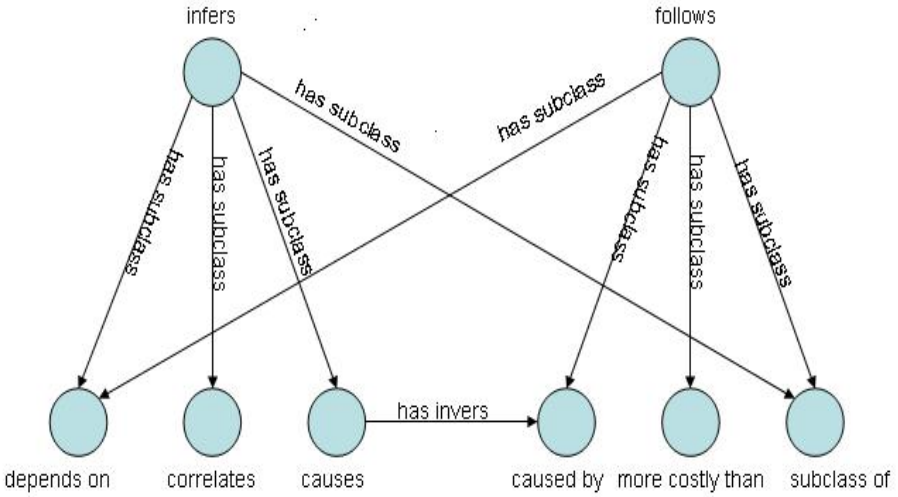

Figure 2.2: The semantic relation hierarchy

Knowledge intensive CCBR

By using dialog inferencing and question ranking it is possible to lessen the cognitive load on the user, and to have the questions asked in a more natural way for the users you can implement consistent question clustering and coherent question sequencing [10].

After the initial input or after a question has been answered by the user, the system will match the updated query with the cases in the casebase and retrieve a selection of the best matching cases. Then the feature instances present in the retrieved cases, which are missing in the query, will be turned into questions with answers. The system can then try to infer new information by using the relations between nodes in the domain model. This is done by activating the query features corresponding nodes in the domain model and explore the relations from each of these nodes. If you have a relation which can adhere to the infers relation, as seen in figure 2.2, the system will be able to construct explanation paths through these relations to nodes, corresponding to the answers to the questions, with a weight. This weight is the relation weight between the two nodes or the multiplication of the relations weights if the system needs to traverse several nodes and relations, see figure 2.3. Finally in what is called the focus step, the system evaluates the weights on the explanation paths against a threshold to decide if the information in

the destination node should be added to the query as a new feature.

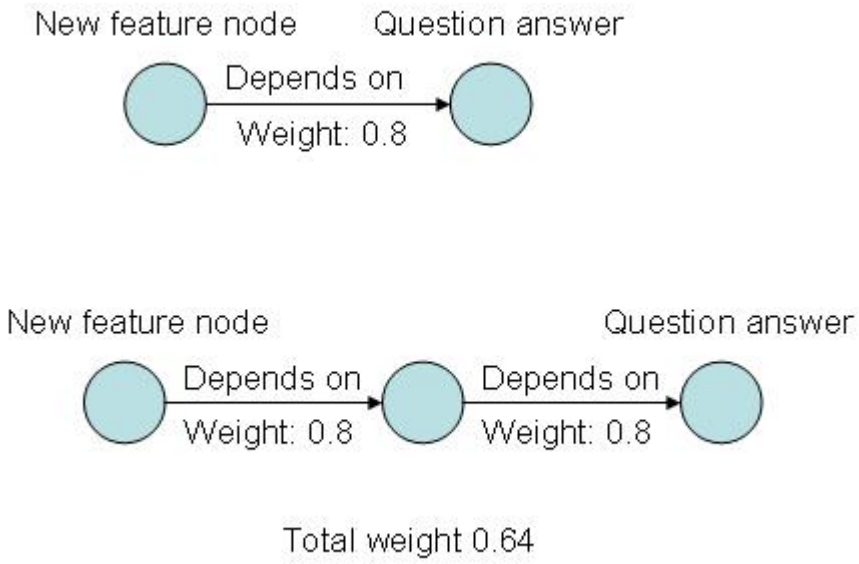

Figure 2.3: Explanation paths with their weights

The remaining questions will then be ranked. This is done by applying a method similar to the one used for inferencing, but using relations that can adhere to the follows relation instead. Among the remaining questions, the system will activate their nodes in the domain model and try to build explanation paths between them. Then these paths will be evaluated according to their relation weight in comparison to a threshold. The questions with an explanation path that justifies being asked before another question according to the threshold is then placed in one list, the rest is placed in a second list. The first list is then ranked by the weights on the explanation paths and added to the top of the question list presented to the user. The second list of questions can then be ranked by using statistical metrics such as information gain or occurrence frequency and added to the bottom of the list presented to the user.

Consistent question clustering is to group up questions within the question ranking so that questions with a semantic relation between the features are close together. This is to make it easier on the user and make it more natural, because the user will be able to see groups of similar questions instead of sin-

gle ones spread out. As an example it would be preferable to have questions about what illness the patient is experiencing grouped up, then questions about what dosage the patient is taking and for how long it has taken the drug and so on, instead of having the questions spread out as one question about experienced illness, then dosage and then back to experienced illness and so on.

Coherent Question selection is about making the question sequence more natural. This is done by implementing that in the next question sequence, after a questions has been answered by a user, the questions asked are ranked so that follow up questions from the last answer comes high up on the question ranking. If a question about a feature were answered by the user and this feature has one or more subclasses it would be natural to ask about this or these subclasses next, so these questions should be ranked high in the next question sequence.

Taxonomic CCBR

Earlier Gupta made a solution to lessen the cognitive load on the user with Taxonomic CCBR [14]. In Taxonomic CCBR the method is to create a taxonomy of nodes placed in a hierarchy, with the nodes representing question answer pairs. The relations between these nodes are "is a" relations or in other words abstraction, which means that an ancestor node will subsume all its children nodes. Cases consist of a solution and a problem description made up of features, which are question answer pairs. These features index the cases to the taxonomy and no two features in a case refer to the same question or are related by an abstraction relation. Each of these indexes also have a similarity score associated with them. This score is calculated by looking at where the answered feature in the query are, compared to the feature indexed by a case in a taxonomy. If the indexed feature is an ancestor of the query feature the score is 1. If it is a descendant it is lower than 1 and will be lower the further down the hierarchy it is, although it also considers the density of nodes. For a parent node this score will be accumulated from all its children indexing scores and combined, shown in figure 2.4.

A set of cases is selected from the casebase and presented to the user between each dialogue cycle. This is based on the similarity of the query from the user compared to the cases in the casebase and only the cases whose similarity

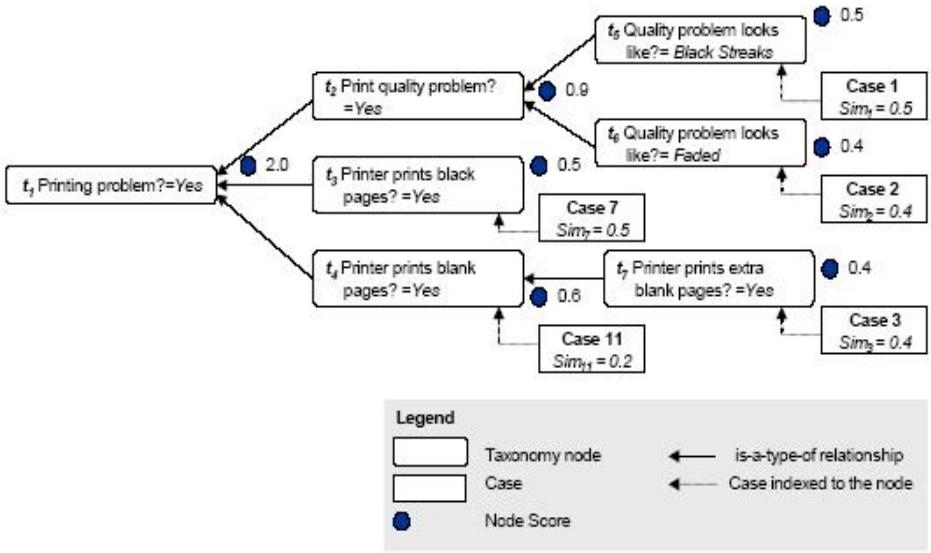

Figure 2.4: Example of taxonomy with indexing from cases and scores, from [14]

exceeds a threshold are retrieved. With these taxonomies and indexed cases we can select questions to ask and rank these. In the figure 2.4 it is assumed that the user has answered the question t1 and the system would then select the descendants directly below this one to present to the user. If the query did not have any answered questions from the taxonomy, the system would choose the most specific node that subsumes the set of retrieved cases. To rank the questions the system gives each question two scores, taxonomy score which would be 2.0 in this example and equal for t2, t3 and t4, and a score that is the sum of its corresponding question answer pair scores, in this example 0.9 for t2, 0.5 for t3 and 0.6 for t4. The questions are then ranked first by taxonomy score and then by the question answer pair score, which gives the list t2, t4 and t3 in this example. They show that these relations reduce the number of questions needed to ask and reduces the number of features needed to index each case.

Aha and Gupta later released an enhanced version, called Causal CCBR [5], where they allow for the use of causal relations between root nodes of Taxonomies.

Chapter 3

System design

When designing a CCBR system there are several parts that needs to be studied and solutions decided upon. In this chapter we describe the design of our system, with case structure, similarity assessment, question creation and ranking, inferencing and the important parts of the core from Trollhetta AS.

3.1 C++ core

The intelligent systems group at NTNU, has used Creek [2] as its CBR implementation system for quite a few years now and with this shift towards CCBR they wish to develop a new system. This project has been started in cooperation with Trollhetta AS and a system core has been implemented during the spring of 2007. As opposed to Creek which was implemented in Java, the new system will be implemented in C++.

There were some bugs with the version that was first made by Trollhetta AS. Tor Gunnar Høst Houeland, one of the four students working on a project this spring, made another version where a lot of these bugs were fixed. This is the version I used in this project.

So far the implemented system is a very simple core without any user interface. You have the possibility of making casebases, adding cases to a casebase, add features to a case and a value to these features, there is also a simple similarity measurement method. However, it has been made so that it can be expanded to include a domain model, as this is something that will

be implemented in the future. This means that every case, feature and value has to be put into a network of entities and relations, which complicates doing minor changes to the system. The semantic network constructed for a case is shown in figure 3.1.

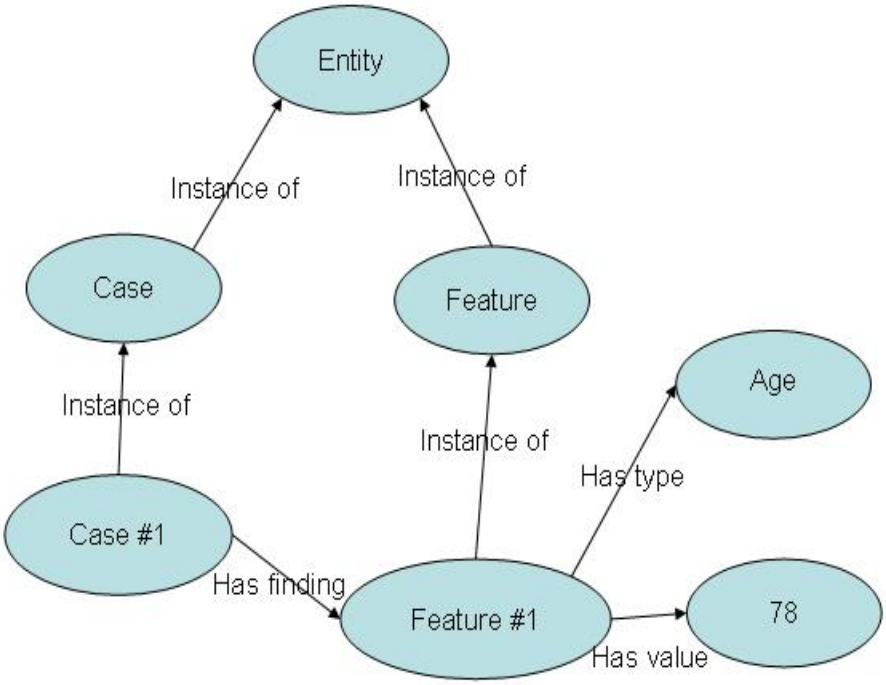

Figure 3.1: The semantic network created for a case

The top level node in this hierarchy is the Entity node. Both Case and Feature is connected to this node through an instance of relation. Each different case and each different feature is then connected to Case and Feature respectively also through an instance of relation. From the different cases you then have a "has finding" relation to each feature which is found in that case, and from each of these features you have a "has type" relation to the correct type and a "has value" relation to the correct value. All values are instances of Value, which again is an instance of Entity and all types are instances if Entity.

Drug	Aspirin Bayer, tablets 500mg
Substance of effect	Acetylsalicylic acid
Dosage	1 tablet, 3 times a day
Period of use	2 days
Experienced problem by patient	Nausea
Experienced problem by patient	Vomiting
Solution	Stopped use of tablets, problems ceased after a couple of hours. Tried other non prescription pain-relieving drugs with different substance of effect, Paracet, Paracetamol.

Table 3.1: Test case 1

3.2 Case design

To make an example of how these cases can look we have made a few test cases, with aid and advice from doctor Arild Faxvaag at St. Olavs Hospital, to show what they will include. It was decided to select drugs from pain relievers listed in Felleskatalogen [1]. We then used the information listed in Felleskatalogen about known side effects, substance of effect and normal dosage, see figure 3.2. We selected a drug and found its substance of effect, then looked at the normal prescribed dosage and chose an appropriate solution within this frame, but making sure we got a few different choices between the cases. Next we selected one or more side effects listed as known side effects of the drug, making sure we chose some similar, but not identical, and some completely different. The solutions were then shortly discussed with Arild Faxvaag to make sure they were reasonable. As there were no data sets available, this was decided as an acceptable solution. Two examples can be seen in table 3.1 and 3.2.

Drug - The commercial name of the drug and also what kind of form it is in and strength if this is applicable. This would be the first thing the patients would input when using the system. Which one to select, if there are several with the same name, should be possible to determine by the patients by looking at the prescription or drug package.

Substance of effect - This is as the name suggests the name of the substance of effect in the drug. The system should be able to infer this from the

C **Aspirin Bayer**

Analgetikum. Antipyretikum. Antireumatikum.

ATC-nr.: N02B A01

Pakningsvedlegg
Varenummer

TABLETTER *500 mg:* *Hver tablett inneh.:* Acetylsalisylsyre 500 mg, hjelpestoffer.
Indikasjoner: Feber, smertetilstander og reumatiske lidelser.
Dosering: ***Voksne:*** 1-2 tabletter inntil 3 ganger daglig med 4-8 timers intervall. Maks. daglig dose er 4 g. ***Barn:*** Hvis nødvendig kan dosen tas inntil 3 ganger daglig med 4-8 timers intervall:

10-15 kg	ca. ⅓ tablett (150 mg)
15-25 kg	½ tablett (250 mg)
25-40 kg	½-1 tablett (250-500 mg)
Over 12 år	1 tablett (500 mg)

Bivirkninger: Mest vanlig er gastrointestinale. Dyspepsi, kvalme, diaré, oppkast og små blødninger kan forekomme. I enkelttilfeller kan blødninger medføre anemi. I sjeldne tilfeller mavesår, i enkelttilfeller med blødning og perforering. Overømfintlighetsreaksjoner: Dyspné og hudreaksjoner kan forekomme, særlig hos astmatikere. Enkeltstående tilfeller av forstyrrelser i nyre- og leverfunksjonen (økte transaminasenivåer), hypoglykemi og alvorlige hudreaksjoner.

Figure 3.2: Example screenshot of the information listed in Felleskatalogen

drug entered by the patient, using the domain model with information from Felleskatalogen.

Dosage - The amount the patient is taking of the drug per day. This can be listed as tablets per day, millilitre per hour and how many times a day and so on.

Period of use - This is to represent how long the patient had used the drug. For shorter periods of use it would be normal to list this in the exact number of days the patient has used the drug, but when the drug is used over longer periods we think that it should be sufficient to round off to number of weeks or months.

Experienced problem by patient - What problem is the patient experiencing, which the patient believes can be related to the drugs the patient is taking.

Drug	Panodil Zapp
Substance of effect	Paracetamol
Dosage	1 tablet, 3 times a day
Period of use	1 days
Experienced problem by patient	Allergic reaction
Experienced problem by patient	Skin rash
Solution	Stopped treatment

Table 3.2: Test case 2

Solution - This is what the patient of the case did to solve the problem, and what effect this action had on the patient. There can also be an explanation as to why this was the selected approach.

After a discussion with Arild Faxvaag about the features we decided to not add the features of age, sex and weight of the patient. These features would probably not be that important in the process of separating cases, except in special circumstances. So for easier demonstration of the system they were removed. However we did discuss how this information may also be imported from other databases if the CCBR system would require the patient to log in, as a lot of this information would already be collected from the patient.

3.3 Similarity assessment

To measure how similar cases are you need a similarity method and the one implemented in todays system is pretty simple. It compares the number of feature value pairs in the query that match the feature value pairs in each of the cases in the casbase and gives a score between 0 and 1, depending on the number of matches out of the total. Where 1 means that you have a complete match and lower score means the case from the casebase does not match all the feature value pairs in the query. The exact formula used to calculate the similarity score is $\frac{1}{distance-1}$, where distance is the number of feature value pairs the case is missing compared to the query. The method is query-biased, which means that you use the features of the query as a base for comparing. So if a query has 3 feature value pairs, any case in the casebase which include those 3 feature value pairs will get a similarity score of 1, any excess feature value pairs does not matter. The value each feature adds to the distance is set to 1 in the system, but this can be set to any

number between 0 and 1 to give different weights to each feature.

One problem with the similarity method in todays system is that it cannot handle several features of the same type in one case. It will only compare the first feature value pair of that type of feature, both in the query and the cases from the casebase. Because of this you can get some unexpected results as seen in figure 3.3. Depending on the order in which the test features are entered you will get different scores on the similarity of the cases. This could cause the patient to be presented with different solutions. By alternating the order in which the features were added to the cases and the query, and then run the system one time for each feature, we were able to circumvent this problem in our test of the cases with the C++ core.

```
test->addFeature("Experienced problem by patient", "Nausea");
test->addFeature("Experienced problem by patient", "Vomiting");
test->addFeature("Drug", "Aspirin, tablets 500mg");
test->addFeature("Substance of effect", "Acetylsalicylic acid");
Case #6 (1)
Case #1 (0.333333)
Case #2 (0.2)
Case #3 (0.2)
Case #4 (0.2)
Case #5 (0.2)
Most similar case: Case #6
Solution to most similar case: Stopped use of tablets, problems ceased after a couple of hours.

test->addFeature("Experienced problem by patient", "Vomiting");
test->addFeature("Experienced problem by patient", "Nausea");
test->addFeature("Drug", "Aspirin, tablets 500mg");
test->addFeature("Substance of effect", "Acetylsalicylic acid");
Case #1 (1)
Case #6 (0.333333)
Case #2 (0.2)
Case #3 (0.2)
Case #4 (0.2)
Case #5 (0.2)
Most similar case: Case #1
Solution to most similar case: Stopped use of tablets, problems ceased after a couple of hours.
```

Figure 3.3: Showing the query statements and how the system is unable to manage two of the same feature in one case

Case 6 only have nausea as experienced problem by patient, while case 1 has vomiting as its first and nausea as its second experienced problem by patient. In the first test nausea is the first experienced problem by patient in the query and since the system only counts the first of a type of feature

it thinks the query has two instances of the value nausea, therefore case 6 match complete with the query. Case 1 on the other hand gets a distance of 2 because the system only sees that it has vomiting as experienced problem by patient. The opposite is true in the second test, here case 1 gets a complete match while case 6 gets a distance of 2.

3.4 Domain model

Building the domain model has not had primary focus yet, but it has been considered necessary and very valuable to the system, as a domain model is needed to use many of the methods for question selection in section 2.2.1. When looking at the domain model I have focused on what will be required to make the knowledge intensive methods work and what other possibilities it offers. With the system today all cases, features and values are put into a knowledge model and connected through a network, but without connecting this network to a domain model it is of no use.

An example of how this domain model could be implemented is shown in figure 3.4. Here we can see how the different drugs can be organized in subclasses by which substance of effect they have. With a "depends on" relation from these subclasses to the appropriate instance of substance of effect, it will be possible to infer what substance of effect a patient is getting when we know the drug. As seen in figure 3.4 if we know that a patient is taking Aspirin the system will be able to infer that the substance of effect is acetylsalicylic acid, because aspirin is an instance of drugs with acetylsalicylic acid, which has a depends on relation to acetylsalicylic acid, which again is an instance of substance of effect.

It is also shown how side effects can be connected to the different instances of drugs by the causes relation. In figur 3.4 we have that Aspirin can cause skin rash, nausea and vomiting. These relations will have to be weighted according to the probability of this effect happening to a patient, as the drug will not always cause these side effects. We also see the possibility of placing a depends on relation between side effects where this is appropriate, as in vomiting depends on nausea.

It would be helpful to be able to import information from other databases,

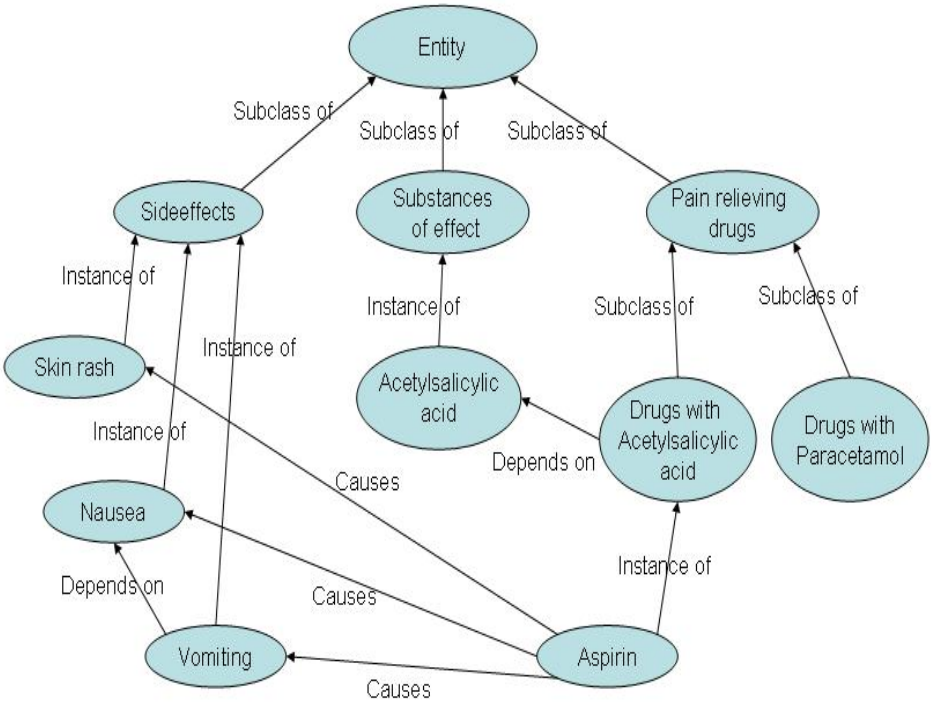

Figure 3.4: Example showing how part of the domain model should be

preferable automatically, into the domain model. With these tools it should be possible to design a general domain model where you have the drugs from Felleskatalogen, with their substance of effects, their side effects and so on. Also importing patient data, such as age, sex, weight, height and so on, from patient records if needed.

3.5 System process

In a CCBR system, the system cycle that runs the process forward is the interactive dialogue with the user. Questions are generated by turning feature value pairs, from cases in the casebase that do not match the feature value pairs in the query, into new questions. A typical example could be that you have a case in the casebase with a feature "Experienced problem by patient" and value "Nausea". If the query do not have that feature value pair, a question added to the question list could be "What problems have you experienced after using the drug?", with "Nausea" as one of its answers.This

can of course become quite a large list with a large number of different cases in the casebase, so we try to remove unnecessary questions and rank the remaining to lessen the cognitive load on the users. From section 2.2.1 there are several methods to improve this.

3.5.1 System initiation and inferencing

To start of the dialog the patient would input the drug he or she is using. Preferable this would be done from a list so the patient do not have to type out anything themselves, just select the correct options throughout the system interaction. As it will be very important to know which drug the patient is using to narrow the amount of cases that needs to be searched, this should always be answered first. This is because if the goal is to include most of the drugs in Fellekatalogen one day, there will be a very large amount of cases with similar side effects, but for different drugs and there will of course be different solutions of what the patient should do depending on what drug the patient is using. Our suggestion is therefore to include this drug selection in the startup of the system, so that this question always will be answered first. This startup procedure could also include the patient logging in to allow for the use of patient records to be imported if this is found useful in the future.

Now the system needs to compare all the cases in the casebase to the query and rank them according to similarity and then select the best matches. This is described in section 3.3. From these cases questions are made as explained earlier in section 3.5. When the system now knows what questions to present to the user it can try to determine the answer of some of them with the help of the domain model. This is done by activating the nodes of the query features in the domain model and follow the relations to explore if it can reach an answer with sufficient weighting to satisfy the threshold. For example if looking at 3.4, if the input drug is Aspirin, the system should be able to infer that the substance of effect for this drug is acetylsalicylic acid and add this to the query. This is because instance of relations normally have a high weight, as does the depends on relation. In this specific case the weights could actually be set to 1 as these relations are 100% sure for these nodes. The causes relations however should not be weigthed high enough for the system to add those nodes to the query, as these are only possibilitys that the drug can cause.

3.5.2 Question ranking and user selection

After adding any features that can be inferred, the system will need to rank the questions left. Finally the system will have to present the highest ranking questions and cases to the patient. Then the patient has to decide if one of the cases presented gives a good enough solution to the problem experienced by the patient, or if not, select one of the questions presented and answer it, starting another round in the cycle. It is of course unlikely that a good enough solution will be presented after the first round, as the patient has not even answered what problem he or she is experiencing.

Question ranking is done by exploring the nodes from the questions features to see if one can create an explanation path, through follows relations, with high enough weight to surpass the threshold. If you as an example have a caused by relation between two question feature nodes, as in A is caused by B, with a high enough weight to surpass the threshold, then question B should be asked before A. Questions that because of this gains ranking over other questions are placed in one list and ranked according to the weight on the explanation path. The remaining questions will at this point be sorted by a simple method that evaluates which question that separates the most cases. This is done by calculating the number of different answers the questions has. These questions are then placed below the ones first separated.

Chapter 4

System simulation

This is a presentation of how the system will be experienced by the user once it is implemented. We have used the core with test cases to show the similarity assessment between the cases for each iteration of the dialogue with the user. To simulate the rest of the system that is not implemented we enter in the features of the query manually for each iteration and show what questions, answers and cases that would be presented to the user.

The user selects the appropriate drug from the list in table 4.1. In this example the user selects "Aspirin, tablets 500mg". The system will find out that the substance of effect for Aspirin is acetylsalicylic acid and add this feature to the query. In figure 4.1 we can see what cases are added to the casebase in this example and at the bottom the current query, with the two features, drug and substance of effect.

Selectable drugs
Aspirin, tablets 500mg
Aporex, tablets
Paracetamol, tablets
Panodil Zapp
Morfin Nycomed Pharma, tablets 10mg

Table 4.1: List of drugs that the user can select from

```
c1->addFeature("Drug", "Aspirin, tablets 500mg");
c1->addFeature("Substance of effect", "Acetylsalicylic acid");
c1->addFeature("Dosage", "1 tablet, 3 times a day");
c1->addFeature("Period of use", "2 days");
c1->addFeature("Experienced problem by patient", "Vomiting");
c1->addFeature("Experienced problem by patient", "Nausea");
c1->addFeature("Solution", "Stopped use of tablets, problems ceased after a couple of
hours. Tried other non prescripton pain-relieving drugs with different substance of
effect. Paracet, Paracetamol");

c2->addFeature("Drug", "Aporex, tablets");
c2->addFeature("Substance of effect", "Paracetamol");
c2->addFeature("Dosage", "1 tablet, 2 times a day");
c2->addFeature("Period of use", "3-4 days");
c2->addFeature("Experienced problem by patient", "Headache");
c2->addFeature("Experienced problem by patient", "Dizziness");
c2->addFeature("Solution", "Contacted doctor and got prescribed another drug. Paralgin
forte");

c3->addFeature("Drug", "Paracetamol, tablets");
c3->addFeature("Substance of effect", "Paracetamol");
c3->addFeature("Dosage", "1 tablet, 3 times a day");
c3->addFeature("Period of use", "2 days");
c3->addFeature("Experienced problem by patient", "Diarrhoea");
c3->addFeature("Solution", "Not known sideeffect");

c4->addFeature("Drug", "Panodil Zapp");
c4->addFeature("Substance of effect", "Paracetamol");
c4->addFeature("Dosage", "1 tablet, 3 times a day");
c4->addFeature("Period of use", "1 day");
c4->addFeature("Experienced problem by patient", "Allergic reaction");
c4->addFeature("Experienced problem by patient", "Skin rash");
c4->addFeature("Solution", "Stopped treatment");

c5->addFeature("Drug", "Morfin Nycomed Pharma, tablets 10mg");
c5->addFeature("Substance of effect", "Morfinsulfat");
c5->addFeature("Dosage", "1 tablet, 4 times a day");
c5->addFeature("Period of use", "2 weeks");
c5->addFeature("Experienced problem by patient", "Reduced urination");
c5->addFeature("Experienced problem by patient", "Itching");
c5->addFeature("Solution", "Consulted physician, decided to continue treatment");

c6->addFeature("Drug", "Aspirin, tablets 500mg");
c6->addFeature("Substance of effect", "Acetylsalicylic acid");
c6->addFeature("Dosage", "1 tablet, 2 times a day");
c6->addFeature("Period of use", "1 days");
c6->addFeature("Experienced problem by patient", "Nausea");
c6->addFeature("Solution", "Stopped use of tablets, problems ceased after a couple of
hours. Tried other non prescripton pain-relieving drugs with different substance of
effect. Paracet, Paracetamol");

c7->addFeature("Drug", "Aspirin, tablets 500mg");
c7->addFeature("Substance of effect", "Acetylsalicylic acid");
c7->addFeature("Dosage", "1 tablet, 3 times a day");
c7->addFeature("Period of use", "2 days");
c7->addFeature("Experienced problem by patient", "Skin rash");
c7->addFeature("Solution", "Contacted doctor and stopped use of tablets, Skin rash
disappeared after one day");


test->addFeature("Drug", "Aspirin, tablets 500mg");
test->addFeature("Substance of effect", "Acetylsalicylic acid");
```

Figure 4.1: Cases added to the casebase and current features added to the query, after drug selection by user

Questions	Answers
What problems have you experienced since taking the drug?	Nausea Vomiting Skin rash
What dosage are you taking of the drug per day?	1 tablet, 2 times a day 1 tablet, 3 times a day
How long have you been using the drug?	1 day 2 days

Table 4.2: List of questions that the user can select from

The run from the similarity assessment of the cases is shown in figure 4.2. The user will then be able to view the best cases or choose among the following questions, with their respective answers, listed in table 4.2. This time the user selects question 2, what dosage are you taking of the drug per day, and answers 1 tablet 3 times a day. This feature value pair are then added to the query, as seen in figure 4.3.

```
Case #1 (1)
Case #6 (1)
Case #7 (1)
Case #2 (0.333333)
Case #3 (0.333333)
Case #4 (0.333333)
Case #5 (0.333333)
Most similar case: Case #1
```

Figure 4.2: Similarity assessment after drug selection

```
test->addFeature("Drug", "Aspirin, tablets 500mg");
test->addFeature("Substance of effect", "Acetylsalicylic acid");
test->addFeature("Dosage", "1 tablet, 3 times a day");
```

Figure 4.3: Query status after first question answered

Questions	Answers
What problems have you experienced since taking the drug?	Nausea Vomiting Skin rash
How long have you been using the drug?	1 day 2 days

Table 4.3: Updated question list

With this updated query a new similarity assessment will be run, with results seen in figure 4.4. Now we see that only 2 cases remain equal to the query. The user will be presented with these cases and the similarity result, together with an updated set of questions. The questions presented are shown in table 4.3 and the user chooses to answer what problems have you experienced since taking the drug with skin rash.

```
Case #1 (1)
Case #7 (1)
Case #6 (0.5)
Case #3 (0.333333)
Case #4 (0.333333)
Case #2 (0.25)
Case #5 (0.25)
Most similar case: Case #1
```

Figure 4.4: Similarity assessment after first question answered

Now the query will have added, as you can see in figure 4.5, the feature "Experienced problem by patient" with value "Skin rash". Again the system will run a similarity assessment and this time only one case matches the query, shown in figure 4.6. This case will be presented to the user, with its suggested solution. The user can of course go back and answer different questions or select other answers, if he or she is not satisfied with the case and solution offered.

```
test->addFeature("Drug", "Aspirin, tablets 500mg");
test->addFeature("Substance of effect", "Acetylsalicylic acid");
test->addFeature("Dosage", "1 tablet, 3 times a day");
test->addFeature("Experienced problem by patient", "Skin rash");
```

Figure 4.5: Query status after second question answered

```
Case #7 (1)
Case #1 (0.5)
Case #6 (0.333333)
Case #3 (0.25)
Case #4 (0.25)
Case #2 (0.2)
Case #5 (0.2)
Most similar case: Case #7
Solution to most similar case: Contacted doctor and stopped use of tablets, Skin rash
```

Figure 4.6: Similarity assessment after second question and solution to the most similar case

Chapter 5

conclusion

5.1 Discussion

The selection of what kind of inferencing and question ranking methods to choose for our CCBR system was relatively easy. It was wanted to use the new CCBR system being developed at NTNU based on the C++ core from Trollhetta AS. As this would be a general CCBR system and not specifically made for this system, it would need to be implemented with a set of general CCBR methods that would work for several systems. As the methods described in [13] satisfies these needs and the needs of our CCBR system, it is natural to continue the work that has been done here at NTNU. Of course we had to check related research and it was very valuable to study Guptas system [14], as it provided another angle on a lot of the same ideas.

On the subject of cases we would have preferred to use real data for the case construction. As this was not available, the solution to create the cases from the information in Felleskatalogen [1] and make up a sound solution, was decided acceptable. As we only used information from Felleskatalogen for the problem descriptions and what the solution presented at this point is not crucial. For a future implementation and real testing of the system, there should of course be collected real data for the cases.

The problem with the similarity method in the C++ core, which can not handle more than one of each type of feature in a case or query, was unresolved and just circumvented in the testing of the cases. This was choice made after examining the code and realizing that we would have to rewrite

most of the similarity method to fix it. As this is a very simple similarity method, that will be further developed and probably completely revised and rewritten in the future, it was decided it was not worth the effort at this time. This was both because of the time constraint, and because it did not have a big impact on the testing of the cases or the simulation of the system.

5.2 Related research

Bichindaritz and Sullivan have developed a system called Care-Partner [8]. This system tutors medical students over the internet, providing them with practice cases to help test their skills. Care-Partner also gives the student a result of either fails to meet standards, adequate and meets all standards.

McSherry is arguing that interactive CBR is a good solution for a lot of sequential diagnosis problems [15]. Also implementing an interface called CBR strategist with demonstration of its use in the domain of computer fault diagnosis.

In "Advancements and trends in medical case-based reasoning: An overview of systems and system development" [16], Nilson and Sollenborn gives a good overview over the field of CBR within the medical domain. They divide systems into 4 groups by looking at their medical application, diagnostic systems, classification systems, tutoring systems and planning systems. They also discuss the CBR specific construction trends for CBR systems within the medical field.

5.3 Conclusion

The study of the CCBR systems in chapter 2 gave a very good insight in knowledge intensive CCBR methods. This gave us a solid platform to build from, when designing our own system. The designed system has cases built from information from Felleskatalogen and we designed the example domain model to meet the requirements of the knowledge intensive CCBR methods used in our system. The designed methods for question selection in our system is inferencing and question ranking. We also described the C++ core from Trollhetta AS and the similarity method present in that code. Finally

we presented how a patient would experience a dialogue with the system.

The design we have created is a good basis for implementing a knowledge intensive CCBR system. The system should work better than a normal CCBR system, because of the inferencing and question ranking methods, which should lessen the cognitive load on the user and require fewer questions answered, to reach a good solution.

There is still quite a bit of work left before this system will be able to be put into effective use, but we have shown some of the possibilities that this system has. The design will need to be fully implemented and a better similarity method for comparing cases should also be studied and implemented. For the cases it will be needed to collect data about solutions used from patients. A complete domain model will also have to be made.

Bibliography

[1] Felleskatalogen. `http://felleskatalogen.no`.

[2] Agnar Aamodt. Knowledge-intensive case-based reasoning in creek. *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004*, pages 1–15, 2004.

[3] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues. methodological variations and system approaches. *AI Communications, Vol 7 Nr 1*, pages 39–59, 1994.

[4] David W. Aha, Leonard A. Breslow, and Hector Munoz-Avila. Conversational case-based reasoning. *Applied intelligence*, pages 1–25, 1999.

[5] David W. Aha, Kalyan Moy Gupta, and Nabil Sandhu. Exploiting taxonomic and causal relations in conversational case retrieval. *Advanced in Case-Based Reasoning: 6th European Conference, ECCBR 2002*, pages 175–182, 2002.

[6] David W. Aha, David Mcsherry, and Qiang Yang. Advances in conversational case-based reasoning. *The knowledge engineering review, vol 20:3*, pages 247–254, 2006.

[7] Ralph Bergmann and Ivo Vollrath. Generalized cases: Representation and steps towards efficient similarity assessment. *Advances in Artificial Intelligence: 23rd Annual German Conference on Artificial Intelligence, Proceedings*, page 698, 1999.

[8] Isabelle Bichindaritz and Keith Sullivan. Generating practice cases for medical training from a knowledge-based decision-support system. *Workshop Proceedings, ECCBR 02*, pages 3–14, 2002.

[9] Mingyang Gu and Agnar Aamodt. *Explanation-Boosted Question Selection in Conversational CBR.* PhD thesis, NTNU, 2005.

[10] Mingyang Gu and Agnar Aamodt. *A Knowledge-Intensive Method for CCBR*. PhD thesis, NTNU, 2005.

[11] Mingyang Gu and Agnar Aamodt. *Supporting Generalized Cases in Conversational CBR*. PhD thesis, NTNU, 2005.

[12] Mingyang Gu and Agnar Aamodt. *Dialog Learning in Conversational CBR*. PhD thesis, NTNU, 2006.

[13] Mingyang Gu and Agnar Aamodt. *Knowledge-Intensive Conversational Case-Based Reasoning in Software Component Retrieval*. PhD thesis, NTNU, 2006.

[14] Kalyan Moy Gupta. Taxonomic conversational case-based reasoning. *Proceedings of the fourth ICCBR*, pages 219–233, 2001.

[15] David McSherry. Interactive case-based reasoning in sequential diagnosis. *Applied Intelligence 14*, pages 65–76, 2001.

[16] Markus Nilsson and Mikael Sollenborn. Advancements and trends in medical case-based reasoning: An overview of systems and system development. *Proceedings of the 17th International FLAIRS Conference, Special Track on Case-Based Reasoning*, pages 178–183, 2004.

Appendices

Appendix A

Test cases

Drug	Aporex, tablets
Substance of effect	Paracetamol
Substance of effect	Dekstropropoksyfenhydrochlorin
Dosage	1 tablet, 2 times a day
Period of use	4 days
Experienced problem by patient	Headache
Experienced problem by patient	Dizziness
Solution	Contacted doctor and got prescribed another drug, Paralgin forte.

Table A.1: Test case 3

Drug	Paracetamol Alpharma, tablets
Substance of effect	Paracetamol
Dosage	1 tablet, 3 times a day
Period of use	2 days
Experienced problem by patient	Diarrhoea
Solution	Not known sideeffect. Contacted doctor.

Table A.2: Test case 4

Drug	Morfin Nycomed Pharma, tablets 10mg
Substance of effect	Morfinsulfat
Dosage	1 tablet, 4 times a day
Period of use	2 weeks
Experienced problem by patient	Reduced urination
Experienced problem by patient	Itching
Solution	Contacted physician, decided to continue treatment.

Table A.3: Test case 5

Drug	Aspirin Bayer, tablets 500mg
Substance of effect	Acetylsalicylic acid
Dosage	1 tablet, 3 times a day
Period of use	2 days
Experienced problem by patient	Nausea
Solution	Stopped use of tablets, problems ceased after a couple of hours. Tried other non-prescription pain-relieving drugs with different substance of effect. Paracet, Paracetamol.

Table A.4: Test case 6

Drug	Paracetamol Alpharma, tablets
Substance of effect	Paracetamol
Dosage	1 tablet, 2 times a day
Period of use	2 days
Experienced problem by patient	Skin rash
Solution	Contacted physician, decided to stop treatment. Skin rash disappeared after a couple of days.

Table A.5: Test case 7

Drug	Aspirin Bayer, tablets 500mg
Substance of effect	Acetylsalicylic acid
Dosage	1 tablet, 3 times a day
Period of use	2 days
Experienced problem by patient	Skin rash
Solution	Contacted doctor and stopped use of tablets. Skin rash disappeared after one day.

Table A.6: Test case 8

Drug	Aporex, tablets
Substance of effect	Paracetamol
Substance of effect	Dekstropropoksyfenhydrochlorin
Dosage	1 tablet, 2 times a day
Period of use	2 days
Experienced problem by patient	Nausea
Experienced problem by patient	Stomachache
Solution	Contacted doctor and stopped treatment. Used lighter pain relievers, Aspirin Bayer. Problems ceased in a few hours.

Table A.7: Test case 9

Drug	Aporex, tablets
Substance of effect	Paracetamol
Substance of effect	Dekstropropoksyfenhydrochlorin
Dosage	1 tablet, 2 times a day
Period of use	5 days
Experienced problem by patient	Skin rash
Solution	Contacted doctor and decided to continue treatment for another 2 days until most of the pain had ceased. Skin rash disappeared after a couple of days after ended treatment.

Table A.8: Test case 10

Appendix B

Included zip file

In the included zip file are the code files of the C++ core version made by Tor Gunnar Høst Houeland, with my added cases and query. As there is no GUI any testing has to be done by changing the query in the code. Changing of the cases or the query is done in the file "old-Main.cpp". It is the possible to run the program to do a similarity measurement. To run the program, a C++ environment is required, EasyEclipse for C++ is recommended. It is possible to get this at `http://www.easyeclipse.org/site/home`. Just create a project and add the files in the zip file.