1

Case-based reasoning in medical image diagnosis

Jo Skjermo

December 17, 2001

[1]NTNU Trondheim
Fakultet for fysikk, informatikk og matematikk
Institutt for Datateknikk og informasjonsvitenskap

Abstract

In the last several years, there has been an increased focus on connecting image processing and artificial intelligence. Especially in the field of medical image diagnostics the benefits for such integration is apparent. In this paper we present use of the Common Object Request Broker Architecture (CORBA), as the mean for connecting existing systems for image processing and artificial intelligence. To visualize this, we will use CORBA for connecting Dynamic Imager and JavaCreek. Dynamic Imager is an image processing software, that is especially suitable for setting up and test customized sequences of image processing operations. JavaCreek is an artificial intelligence software based on the Cased-Based Reasoning (CBR) theory.

After connecting the two software systems with CORBA, we proceed develop the specific image processing methods for data gathering, and a knowledge base for diagnosis in the artificial intelligence system. The image processing methods and the knowledge base are produced for one special knowledge domain, for visualizing how the proposed system can help in medical image diagnostics.

The task we use to visualize our approach, is detecting malignancy in breast tumors, from magnetic resonance (MR) images taken over time as contrast agents is injected. This is from a reasonable new method for deciding if a tumor is malignant or benign. All image processing methods and the knowledge model is produced to let the two system cooperate to find and diagnose tumors.

The image processing methods, the knowledge model, and the selected software with the CORBA connection, was the basis for our system implementation. The implementation was tested with data gathered during the development of the clinical method for determining if a tumor is malignant, from the MR images. In all 127 patient cases was available, where 77 has malignant tumors in the gathered images. The results was then compared with diagnosis methods based both on manual detection, and on other image processing methods. Although the found results were promising, there was also found several areas for future work.

Acknowledgement

When I approached the department of Computer and Information science with my ideas for this paper, amanuensis Ketil Bø and professor Agnar Aamodt both offered to help and guide my work. It was Ketil Bø that introduced me to digital image processing, and Agnar Aamodt that introduced me to artificial intelligence in the first place. Both helped tremendously with defining and writing this paper, and directing me in the right direction.

Parts of the work described in this paper utilized the JavaCreek software. Fellow student Frode Sørmo, based on works from Aagnar Aamodt, produced the JavaCreek software. Frode Sørmo was extremely helpful in the process of utilizing JavaCreek to its fullest extent, and explaining the finer points in the use of this software.

The Dynamic Imager software was used in the work described in this paper. The company Cetroon produces this software. The employee at Cetroon was always helpful in suggesting improvements, or tracking down bugs.

For the medical part of the work described in this paper, I had good help from Geir Torheim and Kjell Arne Kvistad from the Regional Hospital In Trondheim (RiT). Both Geir Torheim and Kjell Arne Kvistad has been highly involved in develop the medical methods this paper utilizes, and has help both in describing the works done until now, and also helped with suggestions for improvements.

Finally, I wish to thank my family and friends for all the support during the process of writing this paper.

Contents

List of Figures

Chapter 1

GOAL

1.1 Introduction.

This thesis consists of seven parts plus appendices. Each part corresponds logically to the stages involved in building the thesis. In the first part we revive the goal of this thesis, and any problem statements. In the second part we take a closer look at the background and motivation for this thesis. In the third part we develop our method and framework for approaching our goal. Part four is a closer look at earlier research our approach builds upon. In part five we go from our framework, towards an implementation of an experimental system for approaching our goal. In part six we take a closer look at our results, in comparison with previous results. In part seven, we sum up any findings and conclusions.

1.2 Main goals and thesis requirements.

In this thesis we will take a closer look at use of image diagnostic and artificial intelligence for diagnosing medical images. The basic material will be medical images where time is of importance, more specific, Magnetic Resonance (MR) Images of female breasts, taken while a contrast agent is injected.

From these images we will utilize methods from pattern recognition and digital image processing to extract data from these images. The data will then be used as a starting point to determine a diagnosis, by utilizing methods from artificial intelligence, and specifically case-based reasoning (CBR). Both general domain knowledge and case specific knowledge will be used.

As the images utilized in this thesis are from a study of breast cancer, the diagnosis we will be working towards, will be a determination if an image sequence contains a malignant tumor, or not. A system solution for determining a diagnosis must be specified, and an experimental implementation of the system solution must show essential parts of the system, and produce sufficient data to compare results with earlier results.

To approach a solution to these requirements, the digital image manipulation software "Dynamic Imager" will be used for pattern recognition and to get the acquired data to a form that can be utilized by an artificial intelligence system. The artificial intelligence software "JavaCreek" will be used to find a diagnosis in the required form, from the provided data.

1.3 Sub goals.

From our goal and thesis requirements, we find at once that we have to introduce at least three major sub goals that must be defined and solved:

Sub goal 1: As the available data is in the form of MR images, these datas must be translated into a form that can be understood and manipulated

by the JavaCreek CBR system. By the goal description, Dynamic Imager should be used to manage this.

Sub goal 2: For the JavaCreek CBR system to produce a diagnosis from the data made available to it, the available general knowledge and earlier experiment of the domain must be incorporated into the system. This knowledge forms what is known as a "knowledge base".

Sub goal 3: After Dynamic Imager has produced data in a form that can be utilized by JavaCreek; the data must be made available to JavaCreek from Dynamic Imager in some way. The third subgoal is therefore to find a way to give JavaCreek the data from Dynamic Imager, and start JavaCreek in the process of finding a diagnosis. As Dynamic Imager utilizes C++, and JavaCreek Java, it is not a trivial problem to make this connection in a non-manual way.

These three sub goals from now on form the basis for our approach to solve our primary goal of producing diagnoses from the MR images.

Chapter 2

BACKGROUND AND MOTIVATION.

2.1 Introduction.

In this part we will take a closer look at the background and motivation for this thesis. We will explain our motivation both from a local point of view, as well as from an international, "state of the art", point of view.

2.1.1 A real world problem.

At the Regional Hospital of Trondheim (RIT), Norway, there is a department of Radiology. This department has in the last years had an ongoing study of utilizing MR imaging as a non-invasive method to diagnose if tumors in breasts are malignant or not. Up to 127 patients have participated in this study, after detection of a tumor. In this study, most evaluation of the MR images has been conducted manually by pairs of physicians, with exception of some semi-automatic preprocessing with the "Dynalize" software, developed internally at the department. A short overview of some findings with special interest for this thesis, and the approach and results can be found in chapter 3.

2.1.2 System introduction.

"Dynamic imager" is software for fast development of digital image processing. The software is based upon the idea of modules. Each module is a Dynamic Linked Library (DLL), and can communicate with other modules. Dynamic Imager is therefore in all essence a graphic interface for connecting, controlling and reusing DLL's for digital image manipulation. The software comes with many standard modules for digital image manipulation and segmentation, but it is also quite easy to produce new modules using the C++ programming language. "JavaCreek" is a CBR system written in the Java programming language. In essence JavaCreek is artificial intelligence software, where previous experience is stored, together with general knowledge of the domain from which the experience originated from. The software can then be used to solve new problems in this domain by utilizing both earlier experience, and general knowledge. For a closer introduction to CBR and JavaCreek see chapter 3.

2.2 Our motivation.

This is in all essence a work where we contribute towards a system that can understand medical images. Contributions towards such systems do exist from before, and there is already much work done in this field. Why then is this of interest for us?

The reason for our motivation is:

- Internationally, the research field of malignant tumor detection in breasts is of growing interest, as early detection and diagnosis is of vital importance for the patients in question. By producing a semi-automatic computational system for decision support in diagnose malignancy, based on the research done in the study at RIT. In this way we hope we can contribute towards what in the far future may become a fully automatic diagnosis system for detection and diagnosis of breast cancer, based on artificial intelligence and digital image manipulation and segmentation methods. Also, as the artificial intelligence software we will utilize, is a CBR-based system, we hope that this work also may be of benefit as educational support, and a knowledge repository for this domain.

- Dynamic imager and JavaCreek are both highly available products. As JavaCreek is a CBR-based system, one of the most known artificial intelligence methods available, and Dynamic Imager is a software which enable fast and easily creation of digital image manipulation and segmentation systems, a connecting method for these software is of some interest. This because integrating a digital image manipulation and an artificial intelligence system can open for use in a number of other problems that requires diagnoses from digital image material.

- The JavaCreek CBR system is still under development. As such the work done in this thesis is of interest to this development. As a rather huge amount of data was made available to us from the study at RIT earlier mentioned, this gave a chance to test out JavaCreek on a real world problem where data was available in an abundant amount. At the start of this work, this had not been tested in JavaCreek.

2.3 Contribution highlights.

The main scientific contributions of this thesis are:

- We present a way for connecting together, and utilizing the best from both, digital image manipulation and artificial intelligence, by using only readily available software, methods and standards.

- A "knowledge base" with general domain knowledge for diagnosis of malignant tumors in breasts, based on interpretation of MR images of breasts, for use in the JavaCreek CBR system.

- We present selected image segmentation and manipulation methods for interpretation of digital MR images of breast, taken over time with use of contrast agents, in the context of acquiring information on suspected malignant tumors.

- A prototype for a semi-automatic artificial intelligence system for diagnosis of malignant tumors in breasts based on the contributed knowledge base, software connection and image manipulation methods.

2.4 Method of approach.

In this thesis we use an analytical approach for developing a way of connecting the artificial intelligence software and the image manipulation software in question (JavaCreek and Dynamic Imager). During the course of producing this thesis we worked closely with two of the scientists responsible for the earlier mentioned study of detection of malignant tumors from MR images at RIT. Both the work done on modeling the knowledge base for JavaCreek, and finding image manipulation methods for interpretation of MR images in this context were based on selected articles from the mentioned study, and interviews with the two scientists, Kjell Arne Kvistad and Geir Torheim. Based on this work, and the need to utilize the Dynamic Imager and JavaCreek software, we were able to produce a model for our approach, which enabled us to implement our prototype system for testing our model.

Chapter 3

Framework.

3.1 Introduction.

In this chapter we explain the systems, techniques and standards that forms the framework for this thesis. First we take a closer look at case-based reasoning and the JavaCreek system. Then we take a brief tour of the main features of the Dynamic Imager system. We will also introduce the main methods and findings in the work done at detecting malignant tumors from MR images done in at RIT. In the end we describe the CORBA standard for object communication.

3.2 Case-based reasoning and JavaCreek.

Case-based reasoning (CBR) is basically the process of solving a problem by remembering a previous similar situation, by reusing information and knowledge of that situation. This means that CBR is a problem-solving paradigm that differs from other AI methods in many respects. CBR is able to utilize the specific knowledge of previous experienced, concrete problem situation, as well as general domain knowledge. This is possible because CBR does not use only general knowledge of a problem domain for the problem solving.

3.2.1 The CREEK System.

The Creek (Case-Based Reasoning through Extensive Explicit Knowledge) system is an artificial intelligence system originally designed for solving diagnostics and repair problems. The reasoning method used in Creek are Case-Based Reasoning (CBR). This means that a problem is presented to the system as "cases", which can be described as a description of the problem. In the search for a solution to the problem, both earlier introduced cases, and general knowledge of the domain are utilized. If a solution is found for the problem, the solution is made part of the case, and the case can be stored in the system for use in future problem solving. Se Aamodt 1991[1], Aamodt 1994[3] for a closer description of the Creek architecture and design.

3.2.2 The CBR cycle.

The Case-Based reasoning process can generally be described as a cycle as in Aamodt, Plaza, 1994 [4], and this is the main reasoning method in Creek. When looked at from afar, a general CBR cycle may be described by the following four processes:

1. RETRIEVE the most similar case or cases.

2. REUSE the information and knowledge in that case to solve the problem,

3. REVISE the proposed solution.

4. RETAIN the part of this experience likely to be of use in future problem solving.

A new problem is solved by first making, as completely as possible, a new case from the problem and the knowledge given by the problem description. This new case is then used to RETRIEVE one or more previously experienced cases. The new cases and the case found in "retrieve" is then REUSED in the process of trying to find a solution for the problem. This solution is then REVISED. If the solution is found to be acceptable, the new experience is RETAINED by incorporating it into the existing case-base (knowledge repository).

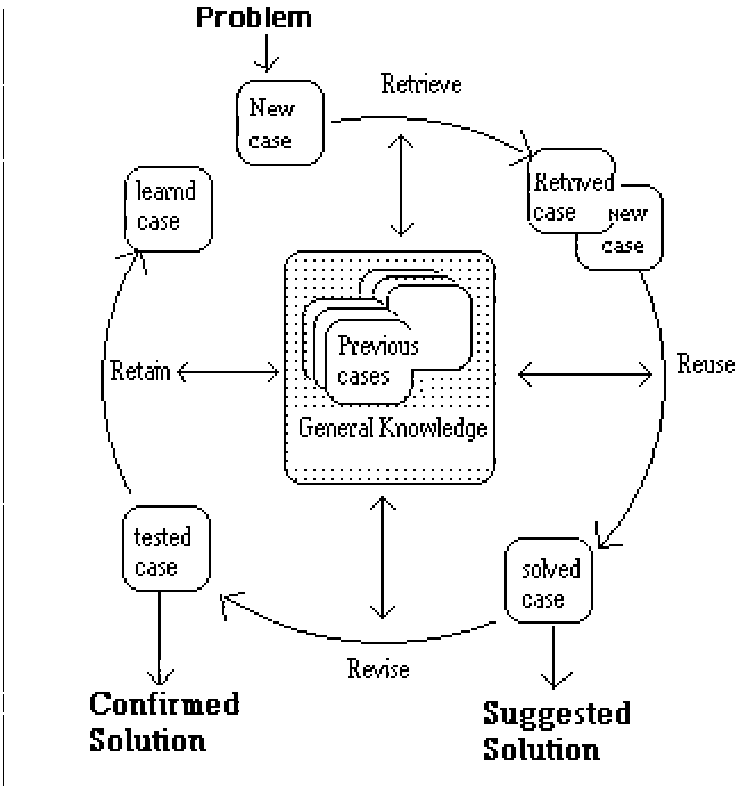

Figure 3.1: Figure of the CBR cycle, reproduced from Aamodt, Plaze 1994 [4].

3.2.3 Creek and JavaCreek.

JavaCreek, described in Srmo 2000 [12] is a CBR implementation based on the Creek system, and the CreekL implementation. In JavaCreek (and CreekL) the knowledge representation is based on frames (Minsky 1975[11]), where knowledge is represented in a semantic net. In this net, the semantic meaning of entities is interpreted from their relations to other entities in the net. An entity and the relations to and from it, defines a frame.

In JavaCreek a case is represented as a type of frame. At the start the case will have related findings, as well as goal and solution constraints. As the case is utilized in the CBR cycle, derived findings and possible solutions, as well as explanations for why these where activated, is added to the frame. If the case is solved, it will also have a solution/diagnosis, and an explanation for how it was derived, contained in the frame.

The JavaCreek and CREEK system is based on an integrated model of problem solving and learning in a computational system. As such JavaCreek is an Explanation-Driven CBR system, which is especially suitable for problems in open and weak knowledge domain, and where at a detailed level a goal is to find the explanation for why a diagnostic hypothesis should be preferred over another, and not only to find a diagnosis.

In Explanation-Driven CBR system the primarily role of general knowledge is to produce explanations to support and control the case-based process. A generic mechanism, called the "explanation engine" constitutes the four fundamental reasoning methods in the CBR cycle. Each of the four reasoning methods is split into the three subtasks ACTIVATE, EXPLAIN, and FOCUS - as illustrated in 3.2

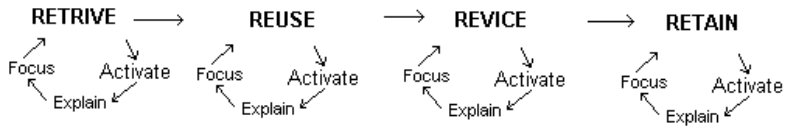

Figure 3.2: Figure of the explanation engine.

The methods in the explanation engine operate briefly as follows:

- ACTIVATE takes the present goal and situation description, and generates a set of concepts suggested as relevant for further processing. This method relies heavily on earlier cases.
- EXPLAIN builds support for the concepts suggested by ACTIVATE. This method relies heavily on the general knowledge to sort the relevance for each of the concepts.

- FOCUS makes the final selection among the competing concepts, when needed, and does so by, among other things, utilizing information of the reasoning goal, and possible constraints.

For a more in-depth explanation on the ACTIVATE-EXPLAIN-FOCUS cycle, see Aamodt 1993[2].

3.3 Dynamic Imager.

Dynamic Imager (DI) is a multithreaded image processing visual experimenting environment. It is specially designed for easily setting up and test customized sequences of image processing operations. DI simplifies the developing process by supplying the developer with a lot of the tools needed in developing sequences of image processing operations, and is specifically targeted towards scientists needing a fast experimental environment to analyze their data. This chapter is deduced from the DI user manual. More information about DI can be found at [5]

3.3.1 Modules.

In DI, a "module" is a standalone entity with parameters to the user or to other modules. See figure 3.3 for an example of a module. Each parameter is of one of the following types:

- Input parameters are a connecting point where other modules may direct data that the module needs. In the user interface, an input parameter shows as an input port.
- Output parameters are where the modules output data is directed. An output parameter can provide data for input parameters on other modules. In the user interface, an output parameter shows as an output port.
- Control parameters are special parameters that enables the user to control and tune whatever task a module is made to accomplish. These parameters are not shown on a module, but rather in DI own interface when a module is selected.

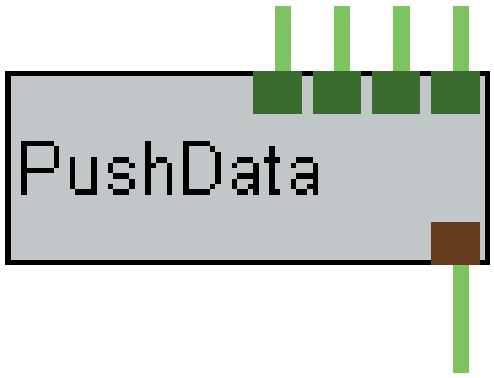

Figure 3.3: Figure of a module with 4 innput, and 1 output.

3.3.2 Main features.

A main feature of DI is that it provides an environment for creating and editing complex sequences of image processing operations, utilizing any number of said operations. This is done by arranging and connecting each processing operation (modules) in a pipelined complex data flow graph (called "module network"). In a module network, output parameters from modules are connected to input parameters of other modules, and such form a graph. Se figure 3.4 for an example of a module network.

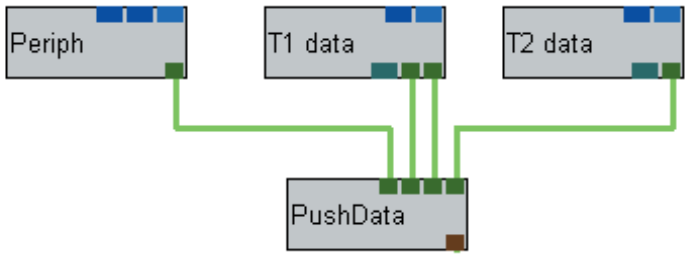

Figure 3.4: Figure of a module network with 4 modules.

Another main feature of DI is the automatic module scheduling. In traditional image processing software the program flow is often manual, and if an error is done early on, stepping back to the point of error is necessary. When the error is corrected, all the steps that were backtracked must be made again. In DI you can at any time make a change to how a single processing task is done, and be presented with the modified output without having to manually re-execute all the following steps.

The simplicity in producing your own modules that you can use in a module network is one of the most important main features if DI. In DI, developers are presented with a rich API for developing their own modules (In the C or C++ computing language). This API gives the developer many functions to use in determining how a module connects to other modules for input or output of data, as well as adding user interface to the modules.

3.4 MR in breast cancer.

3.4.1 Introduction.

Each year more then 2000 women in Norway are diagnosed with breast cancer, and the death rate is approximately 800 a year. This makes breast cancer one of the most deadly malignant diseases for women [9]. It's commonly agreed that early detection is of significant importance for the chance of survival. In this context "detection" is not only detection of a tumor, but also the determination if the tumor is malignant.

A description and a discussion of how to find, and determine if a tumor is malignant with MR, can be found in the doctor thesis of Kjell Arne Kvistad ([6]). The method is based on diagnosis from MR images of the breasts, over a period, while contrast agents affect the outcome of the T1 and T2* MR imaging techniques.

The rest of this section describes how these image dates is used to find tumors, and diagnose any malignancy. The remaining part of this section is divided into two parts. Each part is about data acquisition, and the methods used to detect malignancy in tumors, for two special MR image techniques, named T1 and T2*. For an introduction to MR in general and these two techniques, see 4.2. The section is based on Kvistad [6], Kvistad [8] and Kvistad [9].

3.4.2 T1 data.

In the T1 data acquisition process, 44 images were taken from over the whole breasts. This process was repeated 9 times in short succession. The whole procedure took 59 seconds. After the first 10 seconds, a contrast agent was injected (a process that took 10 second). The result is that the image dates can be viewed as a film over what happens when the contrast agent is injected and spreads through the tissue. This makes visual detection of tumors and ROI's (regions of interests) quite easily, as the contrast agent penetrates a tumor faster then the surrounding tissue, resulting in a higher increase in intensity in pixels in the tumor, than in the surrounding tissues, as the contrast agent spreads. Because of the way the T1 MR images are taken (as described in 4.2, penetration of contrast agent into the tissue, will lead to an increase in the pixel intensity.

T1 signal intensity curve.

In the T1 data set, the average signal intensity in the ROI for each imaging pass, is plotted in a curve. Because the contrast agent increases the pixel intensity in the ROI based on the type of its tissue, the rate and level of increased intensity of the pixels in the ROI will indicate what type of tissue the ROI contains. The form of the curve is then decided to be one of five

predetermined forms, as seen in figure 3.5. The higher the number of the form type, the higher chance for a malignant tumor. This comes directly of how the graph is produced. The more contrast agent in a position, the higher the signal intensity of the corresponding voxel. This gives that a curve of type 5, describes an area where the contrast agent penetrates the tissue fast, before being "washed out" by the blood stream. As malignant tissue is easier for this contrast agent to penetrate and be washed out of then normal tissue, because of increased growth (which need an increase supply of blood), a type 5 graph indicate a malignant tumor to a much higher degree than a graph of type 1. It was found that graphs of type 3,4 and 5 indicated malignant tumors, to increasingly higher degree.

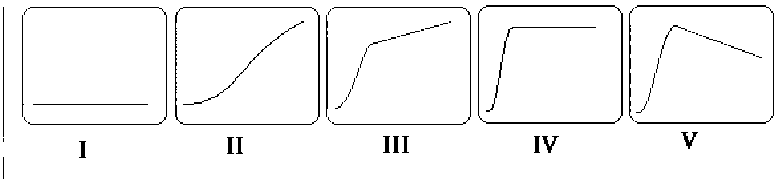

Figure 3.5: Figure over the different T1 signal increase graph types.

Signal intensity increase.

Not only the form of the curve from the T1 ROI data is useful. As the data can be divided in data before contrast agent injection, and after the injection, the difference between the values just prior to injection compared with the value just after injection is of importance. Again the fact that the tissue in a malignant tumor will have a higher and faster penetration of contrast agent than a non-malignant tumor, or normal tissue, was utilized. The higher difference between the signal intensity before and after injecting the contrast agent, the better is the indication that the ROI in question is a malignant tumor. This means that a simple threshold value can help determine if a tumor is malignant. This threshold was set to be at 90% increase in Kvistad [8]. If the found increase in signal intensity was over 90%, it was a good indication that the ROI was indeed a malignant tumor.

Spiculele.

A malignant tumor has by definition a very high degree of growth in comparison with the other tissues in that area, and this growth will be uneven. This is because growth requires constant access to nutrient, which inside the body is provided by the blood. Therefore the growth in a malignant tumor will be higher along blood vessels that the tumor borders to or encompasses.

Because of this uneven growth, the edges of the tumor will appear "jagged", or what is called "spiculele". As of the reason a tumor may have a spiculele shape, the presents of spiculele tumor, indicates that the tumor is malignant. In the work this chapter is based on, the determination if a tumor was of a spiculele shape was done manually by physicians.

Peripheral contrast.

Looking at the peripheral signal intensity of a tumor, in contrast to the tumors center, can also give clues as to determining if the tumor is malignant. If the contrast agent penetrates the tumor faster in the edges, then it does towards the center, it implies a significant amount of new blood vessels in the edge. As new blood vessels imply growth, and growth in a tumor implies malignancy, a high and clear peripheral contrast when the contrast agent penetrates the tumor, indicates a malignant tumor. The determination if a tumor had a high and clear level of peripheral contrast was done manually by physicians.

3.4.3 T2* data.

After a ROI was detected in the T1 data images, the T2* data acquisition was started. The T2* image data is collected only from the part of the chest where ROI was spotted. 40 images was taken in this sequence, and after the first 10 images a rapid injection of contrast agent was administered. Because of the way the T2* MR images are taken (as described in 4.2), penetration of contrast agent into the tissue, will lead to an increase in the pixel intensity.

T2* signal dropp

In the acquired data from the T2* process, the signal intensity in a malignant tumor can be seen to drop as the contrast agent is injected. Taking the average percentage in intensity drop inside a ROI defining the tumor, from just prior, to just after the injection of the contrast agent, will therefore indicate if a tumor is malignant. As in the subsection describing the T1 signal increase, we can use a threshold value to help determine the malignancy. In this case the threshold was set to be 180% in Kvistad [8].

3.5 CORBA

3.5.1 Introduction.

The Common Object Request Broker Architecture (CORBA) is a reference model (CORBA&S) from the Object Management Group [10]. This reference model describes in detail a system for handling of distributed object. As more and more implementations of this model become available, more and more computer languages are supported, where clients and object implementations can communicate regardless of which language they are implemented in, or on which system they are running.

3.5.2 Reference model.

The reference model of CORBA consists of the following components:

- Object Request Broker (ORB): which enables objects to transparently make and receive requests and responses in a distributed environment
- Object Services: a collection of services (objects and interfaces) that support basic functions for using and implementing objects. An example of this is a CORBA naming service that allows lookup of object references.
- Common Facilities: a collection of services that many applications may share, but witch are not as fundamental as the Object Services. For instance, a system management system could be described as a Common Facility.
- Application Objects: which is product of a single vendor or developer. These are not standardized by OMG. This is the top layer of the Reference Model.

In the rest of this chapter we will take a closer look on the ORB and the CORBA naming service, which is an Object Service.

3.5.3 Object Request Broker - ORB.

Basic operation.

Fig 3.6 shows a client sending a request to an object implementation. The client is the entity that wishes to perform an operation on the object, and the object implementation is the code and data that actually implements the objects.

The ORB is responsible for all the mechanisms required to execute the request. This include such things as finding the object implementation,

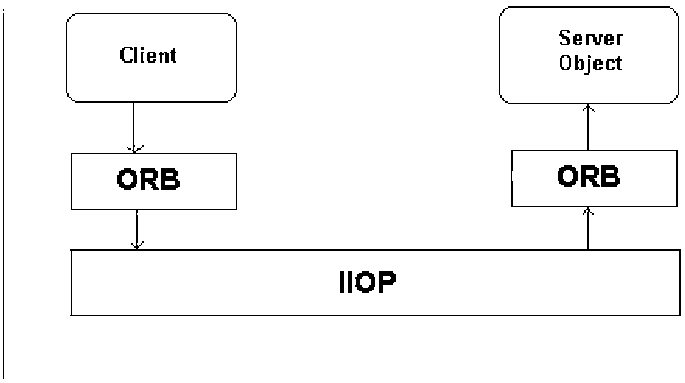

Figure 3.6: Figure of simple corba request.

ready the object for the request, and to communicate the data in the request. In the meantime, the client sees only an interface that is completely independent of such tings as the objects localization, which language it is implemented in and all other things that are not defined in the objects interface. The communication between the ORB of the client and the ORB of the Object is done with OMG's Internet Inter-ORB Protocol (IIOP) standard for communication between distributed objects through the Internet.

IDL Language mapping.

Interface definition Language, defined by OMG, is a language that is used to describe the interfaces to a CORBA object. This is done to ensure a consistent mapping between different computer languages. An example of an IDL definition of an object follows:
interface Echo
String sayEcho(in string message);

This defines an object named "Echo", with the interface "sayEcho", which takes a string input, and returns a string. This IDL definition does not say anything about what "sayEcho" does or how it does it.

Stubb files.

From the IDL definition of the objects, the objects must be translated to objects in the language wanted by the developer. As the IDL do not give any information on how an object will perform the tasks defined by its methods, a translated object only forms a basis for the developer.

Any implementation of CORBA has a language mapper that can translate the IDL to "stubb" files for the language that implementation of CORBA

supports. The stubb files generated for a client and for an object is different, as the stubb files do not only contain the objects description, but also the code needed to utilize CORBA as communication protocol. The developer uses the stubb files as parents for his own versions of the objects on the server side, by using the well-known object oriented approach of inherence. On the client side he uses the generated stubb files as ordinary objects that he can call methods on, and the CORBA code in the object will forward any method invocations to the server objects. The generated code that handles the CORBA communication in the stubb files, are what is known as ORB. That means every object and every client have its own ORB.

3.5.4 The CORBA naming service.

The Corba naming service is an Object Service that most implementation of the CORBA standard has a version of. To understand what a naming service does, we have to look at how a client ORB finds an object.

Interoperable Object Reference - IOR.

In IIOP, the coomunication protocol used in CORBA implementaions, a client finds an object by the objects address in the computer network. This address is known as an Interoperable Object Reference (IOR). The IOR is manufactured by the objects ORB when an instance of the object is generated. If several instances of the same object are generated, each will get its unique IOR address. As any clients that will invoke methods on an instance of the object needs the IOR to find the object, some method must be used to give the client access to the correct IOR address it needs.

In simple applications any instances of the used objects may print the IOR, so the user may give the client the object IOR as innput to the client by hand, but this is not something that is feasible in larger application with a lot of objects and clients.

The Naming service.

Instead of handling IOR's manually, a naming service can be used. A naming service is an Object Service where the IOR of object instances are mapped against names. This means that an instance of a CORBA object can register its IOR into a naming service application, when it starts. The IOR for an instance of a object is registered together with a name. This name used is determined by the developer, and can therefore be known before startup in both the client and server objects. When a client wants to invoke a method in an object instance registered in a naming service, its ORB can find the IOR to use, by simply request the IOR from the naming service by using the name as a lookup key.

Chapter 4

Related research.

4.1 Image analysis.

4.1.1 Introduction.

Image analysis has historically often been divided into two more or less distinct parts, Segmentation and Classification. In segmentation the main objective has been to find areas of similarities (segments) in images, which could signify an object as unique as possible. An example of this is an algorithm that would try to find distinct areas in mammography images. Classification, on the other hand, is the process of taking the areas from Segmentation proceses, and try to say something more about the segments. An example of this could be an algorithm that takes areas found in mammography images with a segmentation algorithm, and try to categorize these segments into the different types of tissues found in breasts.

As in most other cases, we also find that we must do both Segmentation and Classification if we are to succeed in solving our problem.

4.1.2 Segmentation.

As we assume that a domain expert quite easily can detect the basic area of interest in our data, the main Segmentation problem is to fine-tune the area of interest, and to reduce noise in the data. As our data is in the form of a sequence of images from stationary positions in a breast, two old segmentation methods will be of special interest. The metods in question are Motion detection and Region growth.

Motion detection.

In an image sequence, motion is in its simplest form nothing more than a change in the intensities of the pixel values in an image between the pictures in the sequence. This can be compared with what happens when contrast fluid influence tissues in an MR image sequence. One of the simplest approaches for detecting changes between two image frames $f(x, y, t_i)$ and $f(x, y, t_j)$ taken at time t_i and t_j, respectively, is to compare the two images pixel by pixel. One procedure for this is to form a difference image, which may be defined as:

$$d_{ij}\,\{1if(abs(f(x,y,t_i) - f(x,y,t_j))) > \theta\}\,\{0ifotherwise\}$$

A threshold can reduce or totally remove noise. Unfortunaly, the use of a threshold can also remove small or slow moving objects, as these do not attribute much to the difference image. To avoid this problem we accumulate the difference between all the frames comparisons instead of using a threshold in each comparison. An example of this is the AADI method proposed in [13].

In the AADI method the basic idea is to ignore changes that occurs only sporadically over a frame sequence and can therefore be attributed to random noise. In this method an accumulative difference image is formed by comparing a reference image with every subsequent image in the sequence. For each difference between pixels in the reference frame and the present comparison frame, the pixel value in that position in the difference image is increased with 1. In the resulting difference image after this method, the degree of motion in a position is described by the pixel value in that position. The higher the pixel value, the higher degree of motion in that position.

Region growth.

An old and well-known image segmentation method is Region Growth by pixel aggregation. Region Growth is a procedure that groups pixels or sub regions into larger regions. The simplest region growth algorithm is pixel aggregation. In this algorithm a "seed point" is the starting point. From this starting point the neighboring pixels are either added to the region or not, depending if the point in question has similar properties as the "seed point" (or the present region at this time interval as a whole), or not. This is done in a recursive manner until no more neighboring points could be added to the region. If a neighboring pixel has similar properties to the seed point or region, or not, can be determined in many manners. For instance you could use a threshold by which a new pixel value should not difference from the seed point.

Early references on region-oriented segmentation are Muerle and Allaen [14] and Brice and Fennema [15].

4.1.3 Classification.

As our system has a link from the Image Manipulation system to a CBR system, the classification problem here is mostly to translate information acquired from the segmentation algorithms, to a symbolic form that is supported in CBR. Luckily, some of this translation is quite easy, and mostly involves calculating intensity growth or decrease in percentage in a region of interest described by segmentation. However, one crucial translation is to categorize a graph acquired by the previously mentioned translation, into one of five categories. An often used and well-documented method for this sort of classification is to use an Artificial Neural Network, from now referred to as an ANN.

Artificial Neural Network (ANN).

An ANN is based on the use of a multitude of elemental nonlinear computing elements called "neurons" (as seen in figure 4.1), organized, as networks reminiscent of the way in which neurons are believed to be interconnected

in the brain. By successive presentation of training sets of patterns, these nets of neurons can be trained to classify new input patterns on the basis of the training sets. Interest in neural networks started already in the early 1940s, as exemplified by the work of McCullock and Pitts [16], who proposed neuron models in the form of binary threshold and state change between 0 and 1 as a basis for modeling neural system. A network consisting of neurons can be seen in figure 4.2.

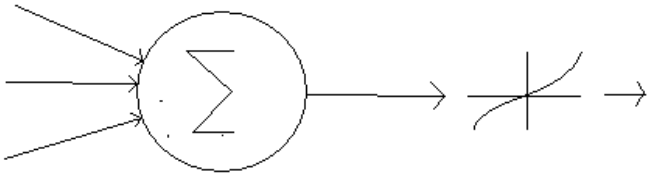

Figure 4.1: Figure of a neuron.

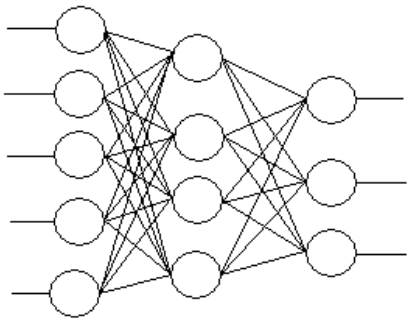

Figure 4.2: Figure of a small neural network, consisting of connected neurons.

Today, in most networks, a neuron is a function where the input is weighted with a weight variable for each input, and where the hard-limiting threshold activation function has been replaced by a soft-limiting "sigmoid" function. Here is an example of a sigmoid function:

$$h(I) = \frac{1}{1 + exp[-(I_j + \theta_j)\theta_o]}$$

Backpropagation in ANN.

In Rumelhart, Hinton and Williams [8], a new training algorithm for multiplayer perceptron nets was described. This method, called "generalized

delta rule for learning by back-propagation" is a quite effective learning algorithm for multiplayer networks. Although there is no way to insure that the training algorithm finds the best solution for a problem, it's assured that it will work towards finding a better solution then it has at the moment.

The basic idea of the generalized delta rule is to find each neuron's contribution to the error for a test pattern, and then correct all the weights that this neurons send output to. The degree that a weight is corrected is in proportion to the error contributed by the nodes that send input to it. In most modern ANN a sigmoid function instead of a threshold activation function. This means that for most cases the generalized delta rule has been rewritten to take this into account.

4.2 Magnetic Resonance.

4.2.1 Introduction.

Magnetic resonance (MR) is a technique used primarily in medical settings to produce high quality images of the inside of the human body. MR is based on a phenomenon involving magnetic fields and radio frequency electromagnetic waves. This phenomenon was independently discovered by Felix Block and Edward Purchell, for which both were awarded the Nobel Prize in 1952. Between 1950 and 1970 MR was developed and mostly used for chemical and physical molecular analysis. In 1971 Raymond Damadian [10] showed that the nuclear magnetic relaxation times of tissues and tumors differed, and therefore motivating the use of magnetic resonance for detection of disease.

Spinn, nucleus and contrast agents.

Some atoms nucleus has a magnetic property called "spinn". Such nucleus will behave as small magnetic fields. This is the basic magnetic phenomenon that MR is based on. When a particle with a net spinn is placed in a magnetic field, its spin will influence the particle to align it to the magnetic field, just like a magnet would. When it's aligned with the magnetic field it is in its low energy state, while it is said to be in a high-energy state when it's aligned in an opposite magnetic direction to the external field. When a particle with spinn is in a magnetic field it can undergo a transition between energy states if hit by a photon, that has en energy that matches the energy difference between the low and high-energy states to the particle.

The hydrogen atom has a nucleus that has spinn. Hydrogen is an atom in abundance in the human body, as the human body is mostly composed of fat and water, both materials mostly composed of hydrogens. The human body is composed of approximately 63 percentage of hydrogen, and therefore MR has historically, primarily used hydrogen nucleus to depict the human body.

Since the midst-eighties, however, scientist started to use contrast agents of Gadolinium to acquire better images. Gadolinium is a rare-earth metal which has very strong paramagnetic properties. This, with its capability to interact with adjacent water protons, gives serious improvements in both T1 and T2* (the major MR imaging techniques used in MR diagnostic imaging).

4.2.2 The T1 MR imaging technique.

In the T1 imaging technique, the particles in the area of the magnetic field are saturated with photons, so to force a translation of enough of the particles to get a balance between particles of high and low energy state (in an saturated state). The time it takes before the particles rearrange to a

low energy state is then measured. As an area contains different tissues, that contains different amounts of particles with spin, the time before a low energy state for the different areas in observation will differ, and this time can be measured. The values gathered for each area can then be used to make a T1 MR image, where each pixel corresponds to a small area. This means that the T1 MR image indirectly is a representation of what tissues the depicted areas consists of.

4.2.3 The T2* MR imaging technique.

When the particles in an area of the magnetic field are in a saturated state, the particles will rotate about the direction of the external magnetic field, in a frequency equal to the frequency of the photon that caused the transition of energy state. However, this rotation will decrease over time because of influence of adjacent particles, and also because of regional differences in the external magnetic field. The time it takes before this rotation to decrease will differ between areas of particles that can be measured, and an MR T2* image can be obtained. A T2* image is therefore, as for T1 MR images, a representation of the tissues the depicted areas consists of, but as the method used to aquire the data was different from the T1 method, the image will also be different.

4.2.4 Clarification.

When we in T1 and T2* talk of small areas, we are actually talking about the smallest areas we can measure the magnetic energy status in. Such an area is called a voxel, and can in most modern MR system be down to 3 cubic millimeters. As T1 or T2* is used, an "average" particle type a voxel consists of is measured, and an image are produced, where the value of a pixel is determined by the particle type the corresponding voxel has most of. For more information on the spin effect, T1 and T2* imaging techniques, and MR in general, see The Basic of MRI [11].

4.3 Feature extraction and classification from T2*-weighted images.

In this section we will take a close look at the result found in a relative new article from Torheim, Godtlibsen, Axelson, Kvistad, Haraldset, Rinck [17]. The article describes different approaches to classify T2*- weighted breast MR images, suggest a semiautomatic ROI definition tool, compares the result with or without noise reduction, and also analize any differences between ROI and pixel-by-pixel based analysis. The T2*-weighted image data used in the study, was the same as used in this thesis. A total of 127 patients with solid breast tumors were examined. Of the 127 patients, 70 was diagnosed with breast cancer, by histopathologic examination, mammography, ultrasound and so on.

4.3.1 Preprocessing.

Before being able to analyze a ROI, the ROI must be defined. The method used to define the ROI is described here. We also discuss noise reduction and the two ways the analysis utilizes the pixels in a ROI.

Noise reduction and Semiautomatic ROI definition.

The noise reduction algorithm used was the *Similar Curves filter* as described in Torheim [17]. In short, similar curves filter, is a smoothing process utilizing only the similar neighbor curves. In the analysis, the selected analysis methods were run on data on which noise reduction was both run and not run.

The region of interest was found by utilizing a region growth algorithm. As the data consists of images taken over time, the data can be seen as a film over the changes in each pixel over time. If normalizing the intensity values to between 0 and 1, and then graph the intensity changes for a pixel, we get a curve over the intensity changes over time.

The starting point for the region growth algorithm was defined as the point where the intensity curve most closely resembled a *reference curve*. The reference curve was made to resemble the intensity curve of a pixel inside a malignant tumor. The comparison was done by calculating the correlation coefficient between the reference curve and the time-intensity curve of interest.

ROI average or pixel-by-pixel based analysis.

The analysis of the image data in this article are not directly based on the pixels, but rather on the curves generated from the pixels. However, as there in this case also were regions of interest, the curve generation can be done by two methods. The first method is the normal method, where a

curve is generated for each pixel. In the other method the whole region of interest is looked on as one area, and the curve is generated by the average signal intensity to all pixels in the region of interest, for each time step. As of these two different methods, the analysis was performed with curves generated from both methods.

4.3.2 Region Of Interest analysis and results.

The article described in this section, try's out several methods for analysis of the ROI's in question. In this subsection we take a closer look at some of the results and finding of the analysis.

The analysis methods.

Several well known analysis methods were tried out. Minimum enhancement threshold, Fisher's linear discriminant function, Probabilistic Neural Net, Error Backpropagation Net and the use of Correlation Coefficient, were tested. The test method used to compare the different analysis and classifying was leave-one-out cross validation. In this test method, each finding is tested against all other findings in turn. For instance, in this case, there were 127 patient cases. For each of the patient cases, the data and known diagnosis of all the other patient cases, were used to define any variables in the classifying method, and in any training needed by sub-symbolic methods used.

Results and findings.

The minimum enhancement threshold was found to be as good as or better classification method than the other methods tested. This can be seen in table 6.1 and 6.2.

Noise reduction and curve production.

The curves used in the analysis methods were all produced from the average intensity in whole ROI's, as this was found to be better then producing curves for each pixel. The reason for this was assumed to be that movement between the image frames, made the values in a position inaccurate. It was assumed that image realignment methods and that a course segmentation to remove the part of the breast that was closest to the lungs (and therefore moved most when the patient breathed), could reduce the errors introduced by movement. However, this was not tried.

The results found by the different analysis methods were all from analysis after noise reduction, as noise reduction improved the results for all analysis methods. As the similar curves filter noise reduction method utilizes only

similar neighboring curves to smooth a pixel, it also reduces the movement noise that was found to highly influence the results.

Chapter 5

Approach and results.

5.1 Introduction.

In this chapter we will explain our approach for meeting the goal of this thesis, as described in chapter 1. We will first present a model for our software, and explain any choices we have taken, with regard to the theories and methods described earlier in this thesis. A system design for the system will then be presented, and explained. From the system design we present an implementation description, and finally we present the results from testing the implementation.

5.2 Model.

5.2.1 Introduction

In this section we will present a model for approaching the goals of this thesis, as described in chapter 1. This is done by extracting the main findings from the framework, and the related research for this thesis, as described in chapter 3 and 4. With these findings in mind, we can then show that our approach for designing a system that will be able to solve the task at hand, is well founded.

In chapter 3 we presented works that described the use of MR images taken over time while contrast agent was inserted, to diagnose malignancy in breast tumors. The data acquisition and the extraction of relevant information from the MR images were in some respect computer aided, but the diagnosis of malignancy in a tumor was done manually. The methods and findings from these works will be the basis for our approach. We will strive to utilize image manipulation methods to make the data extraction from the MR images, and the diagnosis itself, more automatic.

Some parts of our model will already be decided at this point, by the goal description for this thesis that was presented in chapter 1. The goal description states that *Dynamic Imager* 3.3 and *JavaCreek* 3.2 is to be used as the software for image processing and for producing diagnosis. The inner workings of both of these systems will therefore highly influence our approach. Even if our goal description specifically dictates the use of these systems, this is in no way a coincidence. Both of these systems are of the highest quality in their fields, and were designed especially for such use as they are used for in this thesis.

5.2.2 Diagnosis and Artificial Intelligence.

As presented in section 3.2, JavaCreek is a well-suited artificial intelligence system for solving diagnosis problem. It was stated that case-based reasoning system based on Creek was especially good at problem solwing in open and weak domains. An open domain is a domain witch cannot be realistically modeled unless relation between the target system and the external changing world are included, and a weak theory domain is a domain where not all important concept relations are known. Medical diagnosis knowledge domain, are by nature mostly open and weak knowledge domains, and so it is for out knowledge domain. Not all concepts about breast cancer, or how all the known concepts influences the diagnosis is well known, and external knowledge must also be included, as for instance the menstrual cycle of the patient at the time of acquiring the MR data seems to be of some importance.

The use of JavaCreek for the diagnosis part of our software therefore seems well founded. As JavaCreek utilizes both previous cases and general

knowledge for the problem analysis and diagnosis, the main work in utilizing JavaCreek for diagnosis of malignant breast tumors, was to find and generalize any available domain knowledge and the implication connecting the domain knowledge and the cases to any diagnosis.

5.2.3 Image manipulation.

In section 3.3 we took a closer look at *Dynamic Imager*. This software is made specifically for scientists needing a fast experimental environment for image processing. Dynamic Imager provides a number of modules for image manipulation. Another feature of this system is the way the parameters for modules are handled, that effectively removes the need for making user interfaces. But, in our opinion the main feature is the ease to produce new image manipulation modules for the system. Together these features make Dynamic Imager an excellent choice for the software in which we produce and test the image manipulation methods needed to extract data from the available image material.

5.2.4 CORBA for communication

One of the problems in the work described in this thesis was finding a way to connect the system for image manipulation and the artificial intelligence system. JavaCreek is a system implemented in Java, while Dynamic Imager, and user-implemented modules, are implemented in C++. This made connecting the two systems to a non-trivial problem. Java is per definition a programming language that should not try to utilize non-java software, as Java is an interpreted computer language made in a "write once, run everywhere" way. In essence this means that using well known methods for software connection, like native calls should be avoided.

Instead of using methods for software communication that was made for software on one computer, we therefore chose to utilize methods for software communication over networks. As long as the software in question knows the same "language" (protocol), the language in which the software is produced in, does not matter.

There is nothing that says that the different parts of the software must communicate over a network, even if the communication methods are made for that goal. The software parts can easily be placed on the same physical computer. This means that using methods for communication over networks, we can use different computer languages in developing our software, as they speak the same language (protocol) when communicating with parts written in another language. And the different parts of the software can also run on different computers in the network, which may be used in the future to develp a version of the software that supports sutch as loadbalancing or the use of one centralized knowledge domain usable for several remote users at

the same time.

In section 3.5 we described the Common Object Request Broker Architecture (CORBA). As described, when utilizing CORBA, the system developer does nor need to define or implement any protocol for the communication between any clients and servers, as this is already defined.

As of this we see that the use of CORBA can be much easier than, for instance, normal client/server communication using TCP/IP. Client/server communication using TCP/IP is the best known computer communication methods for client/server software, as it is the protocol of choice on the largest computer network on the earth, the Internet. However, use of TCP/IP most often requires the software developer to develop his own protocols that uses TCP/IP as transport protocols, if he can't use some of the already developed protocols, as the well known protocols of HTTP, FTP, SMTP and so on.

Another feature that speaks highly for the use of CORBA for the communication is that it was produced especially for connecting objects, and both Java and C++ (as is the computer languages we utilize), is object-oriented languages. Both of these points indicates that CORBA should probably be a better choice for communication in our instance, than, for instance, normal client/server communication using TCP/IP.

5.3 System design.

5.3.1 Introduction.

In this section we will present the system design for an implementation showing the main feature needed to achieve our goal. First we will present the overall program flow for the system. We then will take a closer look at the building of the knowledge base for JavaCreek. How the knowledge base is built, determines what findings a case will consist of. As the knowledge base will differ for most problem, we will try to define and determine a CORBA interface to JavaCreek that is general enough to be used also for other knowledge bases. However, how to use such an interface will always be influenced and determined by the knowledge model (and especially the findings a case can consist of). We will therefore present the CORBA calling interface in Dynamic Imager in respect to our knowledge model. At the end of the section, an introduction to the image manipulation methods used can be found.

5.3.2 Program flow.

The overall program flow is highly influenced by the fact that Dynamic Imager and the JavaCreek software are to be used. At the start of the work leading up to this thesis, JavaCreek did not have any easily usable user interface for controlling or starting a case-based reasoning cycle with a new case. What was available of user interfaces was a utility for building knowledge bases. During the production of this thesis, this has changed, and new tools with user interfaces have been introduced into JavaCreek, as JavaCreek is still a work in constant change. Nevertheless this did indicate that producing the user interface for our work inside JavaCreek was not our first choice at the time.

As one of the main features in the Dynamic Imager software, was the ease of producing user interfaces to each image module, taking advantage of this feature seemed highly appropriate. Processing modules in DI can be arranged into a module network, where each module by definition can have its own user interface. This gives the opportunity for producing a user interface that highly interconnects with the program flow, as it is the way the modules are connected into a module network that determines the program flow inside DI.

As of this the main program flow for our system is described below. See figure 5.1 for an overview of the program flow.

- In Dynamic Imager: The MR image data for a new case are loaded into the system, and the image processing modules needed to produce any findings needed for building a new case in the case-based reasoning system are activated. Any changes of how any of the modules should

behave, like manual tweaking and corrections, are available in the user interface of each module. The user is at all time presented with the various results of the image processing methods.

- When the user is satisfied with the image processing, he can, from the interface of a module, *inside Dynamic Imager*, activate the building of a new case, and starting the case-based reasoning cycle in JavaCreek. Utilizing CORBA, for calling methods in JavaCreek, does this. Calling methods in JavaCreek using the CORBA interface, will run the case-based reasoning cycle on the new case, and produce a solution/diagnosis for this case. All this will occur in the background, and automatically as soon as the user starts the process.
- When the case-based cycle has produced a diagnosis, the diagnosis is returned to Dynamic Imager through the CORBA interface, and then presented to the user by the user interface of Dynamic Imager.

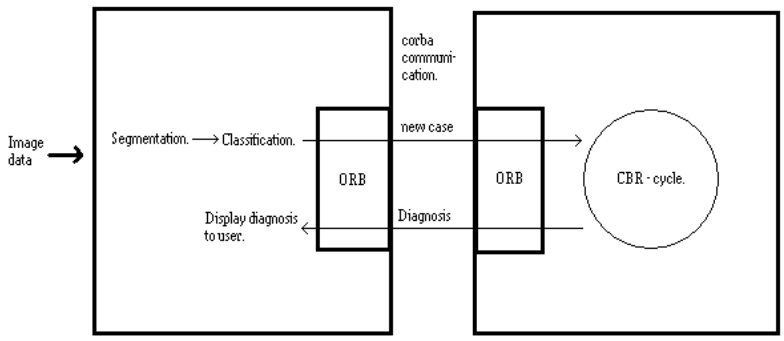

Figure 5.1: Program flow for the system.

As seen in figure 5.1, describing the progam flow, the user will at all time only interact with modules in the Dynamic Imager software. The user can actually to some degree abstract away the fact that it is case-based reasoning with JavaCreek that is the foundation for the diagnosis producing part of the system.

5.3.3 The knowledge model.

Introduction

As stated in section 3.2 , JavaCreek utilizes a semantic net to store cases and the general domain knowledge. In such a net the cases are stored as frames. A frame consists of entities and relations to and from, which form the

basis for the knowledge represented in the net. Each entity has a relation to another entity; which gives the semantic meaning of that entity. This means that building a knowledge model for diagnosing malignancy in breast tumors, can be done by defining relevant entities and the relation between them. In JavaCreek, entities are described by their typical properties, which are inherited by more specialized entities. This means that other entities and their relations to an entity can describe the entity in question. From this we can build our knowledge model by producing a tree of entities and the relations between the entities.

Relation strength.

In the CBR cycle, in the retrieve process, the relations between entities are used as paths to spread the activation of other cases to be examined for similarities. In some cases this leads to the return of several cases that have the same amounts of entities that correspond with the case of interest. In such cases, picking the most similar case to the new one, can be difficult if we have nothing more than the relations between the entities to decide from. Therefore, each relation between entities also gets a strength value associated with it. This value then becomes a part of the general knowledge, and describes which relations have most influence on the connection between frames (read cases).

As of today, the JavaCreek CBR system accepts relation values, although the values are still not incorporated into the CBR cycle. Hopefully, a future version will fully incorporate this concept, so we opt to give all our relation strength values. All our strength values for the relations were statistically calculated from the available patient data, by utilizing normal averaging over the relevant findings of entities where the relations connects them with the same parent entity. For instance, the "si curv" entity (which describes which curve the T1 signal increase are categorized as), is the parent for five other entities, one for each type of curve. The strength of each relation between the "si curve" and the five children entities was therefore calculated by average the number of findings of each type of curve in the patient data. The results of our calculations were then discussed with scientists from RIT, to ensure the correctness.

The tree.

To build the knowledge model we first build the entity tree, and then define the relations and their strengths. We begin with the parent for all the entities, the top node in our entity tree. From this node we divide the entities into three categories. The categories are: "measurable", "observable" and "diagnosis". These are subclasses of the top node in our tree.

The "observable" node has one subclass, the "mens" node. This node

has three value entities associated with it, describing the menstrual capacity of a patient. The value nodes are "premenopause", "postmenopause" and "hrt". The "premenopause" stands for still menstrual active, "postmenopause" stands for a patient which no longer has periods, while a patient with "hrt" is undergoing hormone therapy, and the menstrual cycle can therefore be upset. The diagnosis node has two value entities associated with it, the malignant and the benign entity, which should explain themselves. The entity tree so far can be seen in figuer 5.2

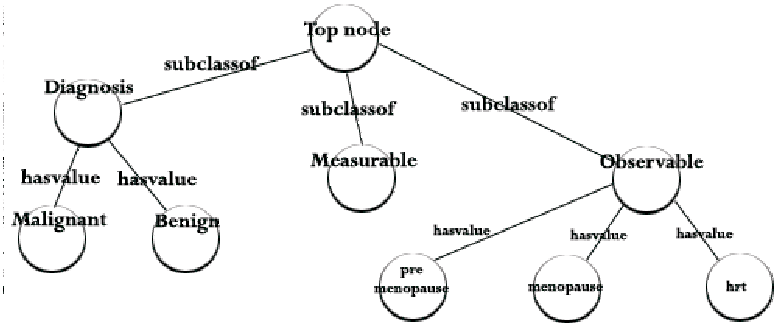

Figure 5.2: Top of the entity tree.

The measurable node has five subclasses which we will discuss separately. These are:

- "si curv", the curve of the T1 signal increase.
- "tissue", the tumor's composition.
- "shape", the tumor's appearance.
- "t1", the signal increase after contrast agent injection in T1.
- "t2", the signal decrease after contrast agent injection in T2.

All of these subclasses correspon correctly with the consepts presented in section 3.4.

The "si curv" node has five value entities associated with it, one for each curve type the T1 data can be described as. These are named "curv1","curve2" and so on. The dividing into the different curves was described in subsection 3.4.2. The "si curv" part of the tree can be seen in figure 5.3

The "tissue" node has two value entities describing if the tumor has higher peripheral penetration of contrast agent, than the tumors center, in the T1 data. The values are "periph" and "noperiph". The concept is described in 3.4.2.

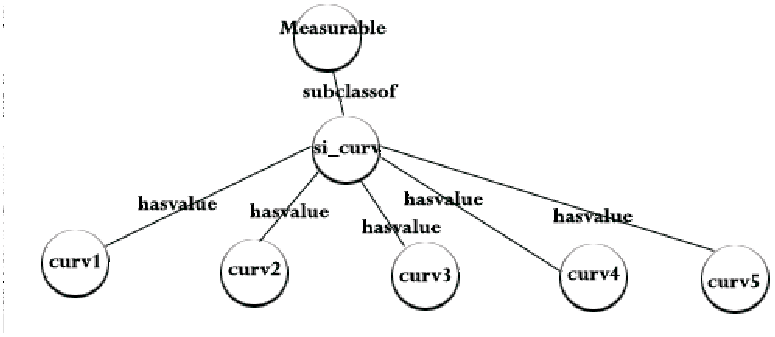

Figure 5.3: The "si curv" part of the entity tree.

The "shape" node has two value entities describing the shape of the tumors in the T1 data. The values are "spic" and "nospic", specifying if a tumor has a spiculele shape or not, as described in 3.4.2.

The "t1" node has two value entities describing if the signal increase after contrast injection in the T1 data, is over a threshold value or not. The values are "highenhance" and "lowenhance", stating that the value was over the threshold or not, respectively. This is closer described in 3.4.2.

The "t2" node has two value entities, describing the signal decrease in a tumor in the T2 data after contrast injection. The values are "highdrop" and "lowdrop". The "highdrop" node states that the signal decrease was more than a threshold value. For more information see 3.4.3.

The tree with all these nodes can be seen in figure 5.4

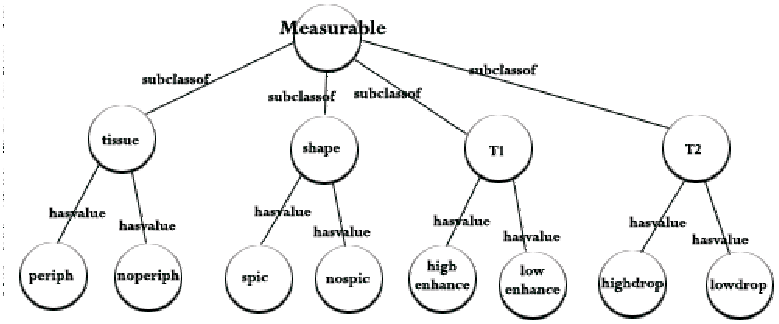

Figure 5.4: The rest of the measurable nodes.

The relations.

From the entity tree we can now describe the relations between the diagnosis entities and all other entities, the value entities. For each value entity in our

tree, we insert a relation to either the malignant or the benign diagnosis entity. All the relations are of the type "Implies", as we cannot say that any value entity with certainty gives a malignant or benign tumor. Each of the "Implies" relations also has an explanation strength set, as described earlier in this subsection.

- For the value entities belonging to the "si curv" node, the "curv1" and "curv2" have an "implies" relation to "benign", while "curv3", "curv4" and "curv5" relate to the "malignant" diagnosis entity.
- For the value entities belonging to the "tissue" node, the "noperiph" has an "implies" relation to benign, while "periph" relates to the "malignant" diagnosis entity.
- For the value entities belonging to the "shape" node, the "nospic" has an "implies" relation to benign, while "spic" relates to the "malignant" diagnosis entity.
- For the value entities belonging to the "t1" node, the "lowenhance" has an "implies" relation to benign, while "highenhance" relates to the "malignant" diagnosis entity.
- For the value entities belonging to the "t2" node, the "lowdrop" has an "implies" relation to benign, while "highdrop" relates to the "malignant" diagnosis entity.
- For the value entities belonging to the "mens" node, the "premenopause" has an "implies" relation to benign, while "menopause" and "hrt" relates to the "malignant" diagnosis entity.

The cases.

From all this we can now describe cases. A solved case in the knowledge model will have a finding from each group of value entities in the tree, and a solution. Here is an example of a solved case:

- Case 1:
  - HasFinding:
  - curv2
  - nospic
  - noperiph
  - lowenhance
  - lowdrop
  - premenopause

- HasSolution:
  - benign
- HasStatus
  - solved

5.3.4 JavaCreek CORBA connection.

Introduction.

In our system design, CORBA was stated as the communication layer between the Dynamic Imager and the JavaCreek systems. Methods in JavaCreek were to be called from modules in Dynamic imager, and the results of any calculations done in JavaCreek from the calls, were to be returned to Dynamic Imager for further calculation and to be displayed to the user of the system. For the Dynamic Imager client CORBA module, this was a straightforward task, as the Dynamic Imager software was produced with such an approach in mind. Each module in Dynamic Imager can themselves handle data presentation to the user, and communicate with other modules. On the other hand, in JavaCreek, there were problems with this approach. JavaCreek was produced to work from user interfaces highly integrated with the logic of the CBR cycle; which produced some extra complexity to our approach.

The JavaCreek CORBA server object

To utilize JavaCreek, a CORBA server object had to be manufactured. This object had to work around any user interfaces, and directly into the main CBR part of the system. In essence this meant that, after stripping away any user interface, an object that interacted with the objects and methods used for controlling the CBR cycle in JavaCreek, just as a user interface would, could be produced. This new object could then be controllable from CORBA. In this context, the JavaCreek CORBA server object can be seen as a "wrapper" and controller for the whole CBR cycle, just as a user interface controller object would. To get a CORBA object that had full control over the CBR cycle for a knowledge model, the methods had to be quite specific in their tasks.

The work that users must be able to do using the CORBA server object for JavaCreek, can in all essence be divided into two major tasks.

- Preprocessing and Case - building. Before anything else can be done, a knowledge model must be loaded. When a knowledge model is loaded, the system is ready for work, and a Case can be produced for use in the CBR cycle.

- CBR cycle: After a new case has been loaded, the CBR cycle can be started with that case. Full control over the RETRIEVE, REUSE and RETAIN parts of the CBR cycle is required. Also, special care for the REVISE part of the CBR cycle has to be integrated, as this is not handled automatically in JavaCreek (se chapter 5 for more information on this).

Each of these two main tasks was again divided into several minor tasks. For instance, each main part of the CBR cycle, was again divided into separate tasks for each of the Activate-Explain-Focus sub cycles. Also the loading of a knowledge model and the insertion of a new case are separate subtasks. In JavaCreek, the program flow and the objects relations between each other are actually the same as this dividing of the tasks. With this in mind, the building of the server object was quite straight forward, as each subtask for the server object can be directly mapped over to the JavaCreek program flow.

The methods in the JavaCreek CORBA server object were therefore set to be as follows.

- Connect, load the knowledge model, and initialize the environment.
- Build the new case to test.
- For the RETRIEVE part of the CBR cycle:
  - a.Run activate.
  - b.Run explain.
  - c.Run focus.
- For the REUSE part of the CBR cycle:
  - a.Run activate.
  - b.Run explain.
  - c.Run focus.
- For the RETAIN part of the CBR cycle:
  - a.Run activate.
  - b.Run explain.
  - c.Run focus.
- Check if the knowledge model with the new solved case should be stored.

Building the JavaCreek CORBA server object from this ensures that the use of the public methods for the server object, inherits the conceptual model of the CBR cycle and JavaCreek.

Extras for data gathering and cross validation.

The data available for this study was mainly data already gathered and studied by scientists at RIT* , and therefore was only available to us in a textual form. We did get one set of image data, but that was clearly not enough for a proper scientific test of the whole proposed system. The textual data was however a full description of the findings, so utilizing these data, we could at least perform a full test of the proposed knowledge model, and the JavaCreek CBR systems performance on this knowledge domain.

To manage this, we introduced another method in the CORBA interface object that came in addition to the methods for normal use of the object. This method was for starting and controlling a "leave one out cross validation" test of the knowledge model with the textual data available. This gave us the opportunities to compare this part of our results with results presented in earlier works in this field. This comparison and some earlier results can be found in 6

Discussion.

As the JavaCreek CORBA server object described follows the conceptual model of how the CBR cycle works, and as the knowledge model to be used is a parameter to the server object from the client, the object will in reality be a universal network (using CORBA) interface for JavaCreek. The method for the cross validation of the patient's data, is on the other hand, especially for this thesis, and cannot be easily used for other knowledge domains.

5.3.5 Image presentation and Dynamic Imager modules.

Introduction.

The client side of the system is a network of modules in Dynamic Imager. The modules in this network read in the data of the T1 and T2* MR image set from a patient case, and processes this data. After the data has been processed, the resulting findings are sent to the JavaCreek CBR system for acquiring a diagnosis and an explanation for this diagnosis, which are returned to the Dynamic Imager for presentation to the user. The communication with JavaCreek are handled on the JavaCreek side by the JavaCreek server Object, while a special module in Dynamic Imager acts as a client, communicating with this object.

The CORBA client module.

The module that performs the communication with the JavaCreek CORBA server object, is a special case when it comes to modules in Dynamic Imager. This module is in essence the client for the JavaCreek CORBA server object. This module works in such a way that it accepts the input for building a case

in a knowledge domain and the place the knowledge model for that domain can be found. The client module then calls methods in the JavaCreek server object, activating first the loading of the relevant knowledge model, and then builds a case with the data available. The server object then returns the status of these operations to the client who then activates the CBR cycle for the case. The client uses the available methods in the server object in sequence, and gets a status report for the result of calling every method.

The form of the case data that the client's CORBA module accepts, is a semicolon-separated list containing the value found, and the relation type this value has. The list can contain any numbers of values and relations. The other input to the client CORBA module, is the filename and location for the knowledge model this case should be run in. This means that this module is in no way usable only for this special knowledge domain and dataset, but is in fact a general module for accessing JavaCreek from Dynamic Imager module networks. To achieve this, there are done no assumptions on which findings a case should contain, or even what knowledge model the case should be run against.

For diagnosing malignancy in breast tumors, utilizing the knowledge model described in 5.3.3, a case will consist of finding according to the knowledge model:

- Has finding value from: mens.
- Has finding value from: tissue.
- Has finding value from: shape.
- Has finding value from: T1.
- Has finding value from: T2.
- Has finding value from: si curv.

The semicolon-separated list sent to the JavaCreek client CORBA module in Dynamic Imager will therefore be of a form similar to the example below:

Has finding;premenopause;has finding;periph;has finding;spiculele;has finding; lowenhance;has finding;lowdrop;has finding;curv3

As for any knowledge model used, the order of the type - value pairs is of no consequence.

In figure 5.5, we can see the CORBA client module, together with the modules providing the data for the module, for our knowledge model.

Tasks in Dynamic Imager.

As in most image analysis software, the task to be done in Dynamic Imager for our problem can be divided into two, segmentation and classification.

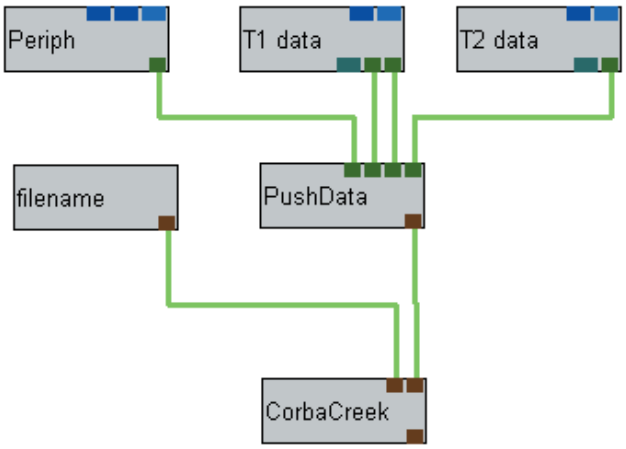

Figure 5.5: The CORBA client module for the JavaCreek server CORBA object, with the modules feeding it with data.

However, the Dynamic Imager modules that deal with classification for this domain are not there to try to diagnose cancer malignancy by themselves, as in most image analysis software. JavaCreek handles the main analysis and classification when it comes to the final diagnosis in this system, as it is a system especially suitable for medical diagnosis tasks. This hopefully reduces the needed complexity of the image classification methods needed, as the task where image classification is needed is simply to get the relevant findings for building a case from the segmented MR images.

The segmentation part of the process is to find and isolate a region of interest defining a tumor in the MR image data set.

To summarize, the tasks that must be done in Dynamic Imager, are, first to find an interesting tumor in the image material by segmentation, and then use classification methods for determining the values needed to build a case for JavaCreek. The process of finding a ROI in the T1 and T2* image datasets is a semi-automatic process in our system. For the T1 image dataset, we first need to find the area in the breast (which slice series) where the tumor is located, while in the T2* dataset this is already done by definition. Then, for both image data sets, the exact area the tumor covers must be defined, before any classification can be done. The classification can be divided into parts corresponding with the JavaCreek case building blocks, except for the menstrual state, as this is a value that the user must insert. In our prototype system the T1, tissue and "si curv" findings are calculated from the T1 image data set, while the T2 finding is calculated

from the T2* image data set. The shape finding, is at present a value inserted from manual inspection. The way we utilized Dynamic Image for our segmentation and classification needs, will be described in the rest of this chapter.

Detection of tumor in T1.

The T1 image dataset is a collection of images taken over the whole breast as a contrast agent is injected. The first task for the system will be to find a breast slice sequence of images where the tumor is highly visible. The injection of the contrast agent makes the detection of the tumor in the images quite easy, as the tumor reacts to the agent. A module for displaying the images one by one, but in a sequence corresponding with the slice, was used for displaying the separate MR images. This means that the user could browse the image sequence of each slice and manually check if the contrast fluid interacted with any tumors. When the user finds the tumor, he marks the image number in the data set for further use. A separate module was made for sorting the images in the T1 data set. This module accepts an input parameter on the image number of interest, and sends out the sequence of the images in the slice of the breast that the tumor is in.

ROI definition

From the images taken over time of a slice (the T2* data set is in that form, while the T1 dataset is in that form after the module just described), we need to define a ROI that covers only a suspected tumor. Region Growth, as we described in chapter 4.1 is one of the oldest and easiest segmentation methods. Because of the simplicity of the region growth method, we decided to use this method if it could produce adequate results. However, the data we had available was not a static image, but a sequence of images that had to be treated as such. This meant that using the region growth method on one image from the sequence would produce a ROI that would not be ideal for all the images in the sequence (as the patient breathes and moves during the MR imaging, and the contrast agent is working over time).

A method for taking the contrast agent into account could be, for instance, to make a new averaged image from all the images in the sequence. However, this would not be a viable solution because the motion between the image frames would make an averaging of pixels produce a smoothing effect in the images. The more motion between the image frames, the more the average image would be smoothed, and much information would be lost. This indicated that we needed to find a method for composing one image from the image sequence before region growth could be applied, that didn't remove too much of the information in the image.

In chapter 4.1, we presented the motion segmentation method. Using

the motion detection algorithm, an image containing only the motion from an image sequence can be produced. The relationship between motion and the image sequences we work with may not be apparent at first view, but after a closer look the connection is quite clear. The Image sequence we work with contains images where contrast agent penetrates the breast over time. Anything that happens over time can be described as a change over time, and so motion can be described as a change. For us this means that there is actually no practical difference in an image sequence of an object moving over the image position, and an image sequence over a tumor's MR resonance changing as contrast agent penetrates the breast. If present, a tumor will be the object in an image sequence that changes most as the contrast agent is injected, this is also the object that will be most visible in the new image after a motion segmentation of the image sequence.

Even after applying the motion detection algorithm, there will still be some noise from the motion and breathing of the patient. This will produce some areas of high contrast in the new images, especially near the lungs. However, the new image after applying the motion detection algorithm can be the base image for a region growth algorithm. The results from first applying the motion segmentation algorithm, and then use the region growth algorithm on the resulting image, could produce a ROI that was quite adequate for our needs.

The region growth algorithm has to have defined a "seed point" inside the region that is to be found. One method for defining the seed point could be to use the point of highest intensity in the image from the region growth algorithm. However, this would not guarantee that the selected seed point actually was inside the tumor, as some of the patient's breathing could induce quite high intensity on some areas, and some of the instruments used trying to reduce motion in the patient had moving parts that showed up in this image. The region growth algorithm as defined in 4.1, also needs a "cutoff" value. For our module this parameter is a percentage value, describing the cutoff from the max and min value found in the ROI (as maximum value in the ROI can vary a lot from one data set to the next, a normal cutoff value would not be good enough). The value is changeable by the user, but for our data set, a value of 23 was found to be quite good.

To insure that the seed point for the region growth algorithm was inside the tumor in our prototype system, we therefore made the user responsible to apply a region covering the whole of a manually detected area that could contain a tumor. This made our method for finding the ROI to actually be a method for better defining the user indicated area of interest. It also introduced the need for a manual detection of possible tumors in the T2* image data sequence. For the module network, the ROI specification now encompasses several modules, depending if the data set is from the T1 or the T2* MR method. For the T2* data set, we utilize some pre-made modules for visualizing the data, and select an area encompassing the suspected

tumor. The image data sequence is passed through a module with the motion detection algorithm, which produce a new motion segmented image. This image, together with the manually selected area, is then sent to a module with the region growth algorithm.

The resulting image from the region growth module is a ROI encompassing the parts of the selected area that changes most during the image sequence (during contrast agent injection). For the T1 data set, the procedure is almost identical. The only difference is that the data set contains image sequences from slices over the whole breast, while we are interested in only the slice where the suspected tumor is best visible. The selection of the slice of best quality is a manual procedure that is simply to find an image where the suspected tumor is visible. The image number, together with the whole data set is then sent to a module we produced for stripping away everything except the image sequence of the wanted slice. This sequence is then treated exactly as the sequence in the T2* data set. The parts of the module networks that handle this for the T1 and T2* can be seen in 5.6 and 5.7. The figures show the module network for the part that handles the data findings for both the T1 and T2* data set.

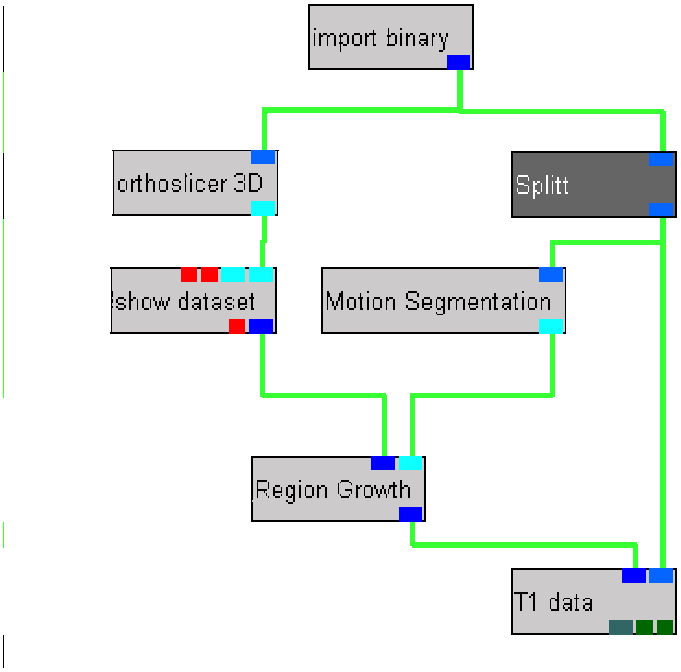

Figure 5.6: Figure of T1 part of the module nettwork.

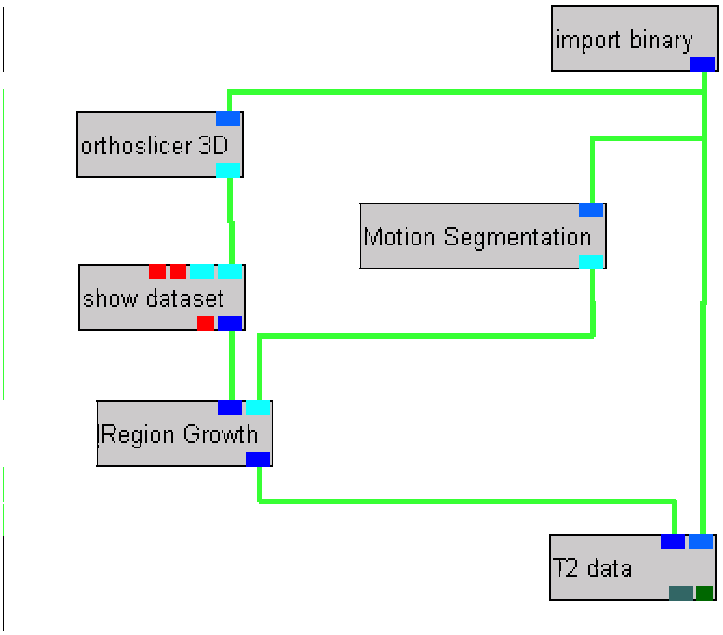

Figure 5.7: Figure of T2* part of the module network.

T1 signal intensity.

From the ROI and the image sequence for the T1 data set, we can now find the T1 signal increase value, and the signal curve, as described in 3.4.2. Finding the signal increase curve was the simple task of average the signal intensity values inside the ROI for each image in the sequence of the selected slice of the breast. This gave a value for each frame in the sequence that forms the signal increase curve.

From this curve we where now able to calculate the signal increase in the tumor after the contrast agent was injected. As the contrast agent was injected during the second MR pass, the signal increase was calculated from the difference between the second and the fourth image in the sequence. This from the fact that during the second pass, the contrast agent had not yet penetrated to the tumor, while during the fourth pass the tumor had been penetrated.

T1 curve categorizing.

Although we have found the signal intensity curve over the signal increase for the T1 data set, it is not in a form that can be utilized by JavaCreek, or the CORBA interface to JavaCreek. In accordance with chapter 3.4.2, and the CORBA interface, we must categorize the curve into one of five types. In

chapter 4, we introduced Backpropagation artificial neural networks. ANN with the Backpropagation learning algorithm is a good method for pattern recognition and categorizing. In its simplest form, a curve can be seen as a pattern, and therefore using a Backpropagation artificial neural network for categorizing should yield good results.

We build a module in Dynamic Imager for this. The module accepted the data values for the curve as input. These values were normalized to values between 0 and 1, before sent through an artificial neural network made especially for classifying the curve type of the signal intensity increase in the T1 image sequence. On startup, this module loads the weights for its network from files, produced earlier during backpropagation training, by a module made especially for the backpropagation training.

T2* signal intensity drop.

From the ROI and image sequence for the T2* MR data, we could calculate the T2* signal intensity drop. This drop was in accordance with chaptersubsect:T2, calculated as the average intensity difference from inside the ROI, between the second image frame, and the lowest intensity value found after frame number four. Again this is the difference between before contrast agent injection, and after injection, as for the T1 intensity increase.

Peripheral difference, spiculele and coding of findings.

In chapter 3, we discussed the difference in contrast agent penetration of the peripheral area of a malignant tumor with regard to the core part. In a module for discovering any peripheral difference, we need to know the difference from a tumor's core part and the peripheral parts. Instead of trying to find a new algorithm for this, we can here reuse the region growth module used earlier. If we reuse the region growth module, but use a higher cutoff value, we produce a ROI covering less of the tumor. A new module was produced, that accepts this new ROI, together with the origianl ROI and the found T1 image sequence. This module uses the ROI's to define what is the core if a tumor, and what is the peripheral area. The module checks if there is faster peripheral penetration of the contrast agent, by comparing the average intensity value for the core and the peripheral parts in the images before and after the injection of the contrast agent. The periperhal module, with the module feeding data to it, can be seen in figure 5.8

The last finding needed for building a case using the CORBA interface module, for this knowledge domain, is the shape of the tumor as described in chapter 3.4. As of today, to define if a tumor has a spiculele shape is a manual task. A user looking at the ROI of the T1 data sequence sees quite easily if a tumor has a spiculele shape (as seen in REF image of ROI). This finding is set in a special module that collects all the other findings,

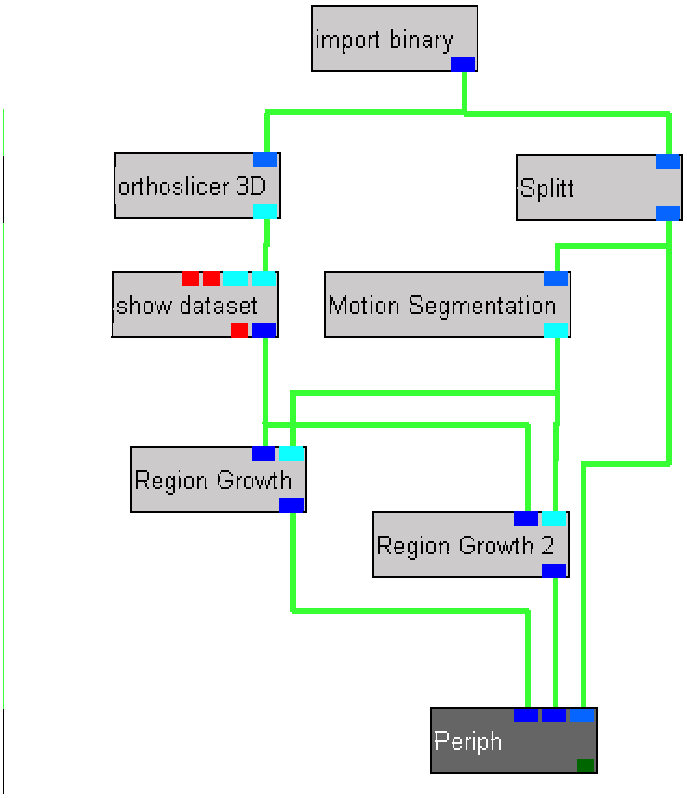

Figure 5.8: Figure of the peripheral module, with the module's feeding data to it.

translates them into the string that the CORBA client module accepts as input, and waits for the user to start the CBR cycle.

Overall module network

In figure 5.9, the whole Dynamic Imager module network can be seen, with displaying modules for all the steps, including displaying modules.

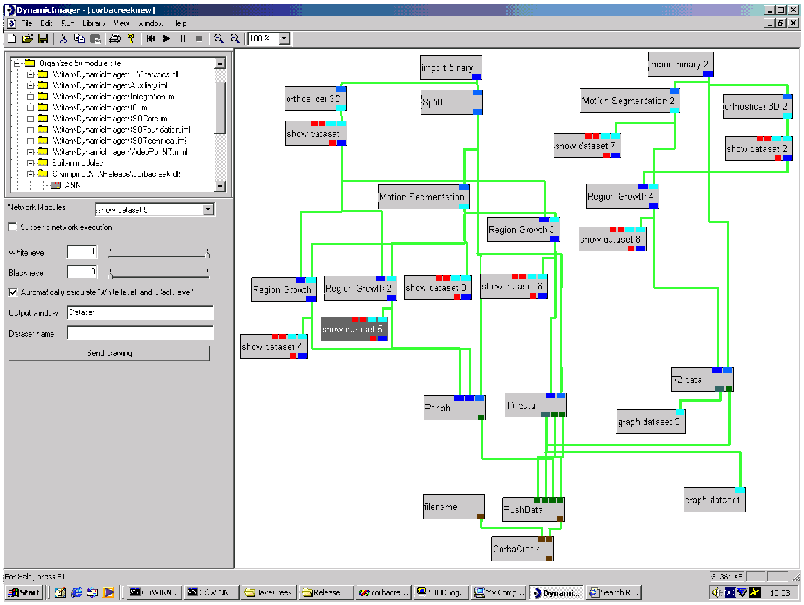

Figure 5.9: Figure of the whole module nettwork.

5.4 Implementation description.

5.4.1 Introduction.

Utilizing JavaCreek, CORBA and Dynamic Imager, we were able to produce a prototype system for our system overview. The production of a knowledge model especially for our problem domain, a general CORBA interface for JavaCreek, and several modules for Dynamic Imager, including a CORBA client module for the JavaCreek server object was necessary. Here we will take a closer look at the implementation of the CORBA interface, and some of the modules we produced for Dynamic Imager.

5.4.2 CORBA communication.

As for all CORBA objects, the implementation of the CORBA server object for JavaCreek and the client module for Dynamic Imager that we described in this chapter is based on an IDL file, as described in chapter 3.4. The IDL file for the task described in this chapter, is shown here:

```
//IDL file for corbacreek, a CORBA interface to JavaCreek.
//It is a part of the JavaCreek package.
module javacreek{
 //This part of JavaCreek is called corbacreek.
 module corbacreek {
  //the interface of the server object is called corbaServer.
  interface corbaServer{

   //Load KM.
   //This methode loads a KM, if no KM is given, load the MR cancer KM.
   //innput: string, path to km.
   void prep(in string KM);

   //Make the working case.
   //innput: name and descriotion of the new case.
   //pre: km must be loaded.
   void prepCase(in string casename,in string casedesc);

   //Fill inn values for the working case.
   //methode for building the case, call for eatch relation-value pair.
   //innput: relation type and its value.
   //pre: prepCase must have been called.
   void ccMakeCase(in string relation,in string value);

   //RETRIVE
```

```
//methodes for the RETRIVE part of the CBR cycle.
//These must be called in sequence.

//calles the retriveActivateResult in JavaCreek.
//pre: the new case must be finnished buildt with ccMakeCase-calles.
void ccRetrieveActivateResult();

//calles the retriveExplainResult in JavaCreek.
//pre: ccRetriveActivateResult must be called.
void ccRetrieveExplainResult();

//calles the retriveFocusResult in JavaCreek.
//pre: ccRetriveExplainResult must be called.
//ret: string with result of the RETRIVE part of CBR cycle.
string ccRetrieveFocusResult();

//REUSE
//methodes for the REUSE part of the CBR cycle.
//These must be called in sequence.

//calls the reuseAcrivateReslut in JavaCreek.
//pre: ccRetriveFocusResult must be called.
void ccReuseActivateResult();

//calls the reuseExplainReslut in JavaCreek.
//pre: ccReuseActivateResult must be called.
void ccReuseExplainResult();

//calls the reuseFocusResult in JavaCreek.
//pre: ccReuseExplainResult must be called.
//ret: string with result of the REUSE part of CBR cycle.
string ccReuseFocusResult();

//RETAIN
//methodes for the RETAIN part of the CBR cycle.

//calls the retainActivateResult in JavaCreek.
//pre: ccReuseExplainResult must be called.
void ccRetainActivateResult();

//calls the retainExplainResult in JavaCreek.
//pre: ccRetainActivateResult must be called.
void ccRetainExplainResult();
```

```
   //calls the retainFocusResult in JavaCreek.
   //pre: ccRetainExplainResult must be called.
   //ret: string with result of the RETAIN part of the CBR cycle.
   string ccRetainFocusResult();

   //For testing with java client, not for public release.
   //calling this methode activates the leave one out cross validataion
   //testing of the knowledge domain for the MR breast cancer domain.
   string testFocus();
  };
 };
};
```

Using the Visipro IDL compilers for Java and C++ from Borland, we could make the CORBA ORB code for the Dynamic Imager client (C++), and the JavaCreek server object (Java). For the server object, the methods were connected with the JavaCreek methods corresponding to the methods. For the Dynamic Imager client the calling of the server objects methods follows the CBR cycle.

The last method in the JavaCreek CORBA server object is for testing out the knowledge model specified earlier in this chapter. When calling the method in question a knowledge model as described in this chapter is made, and the cases of the available patient data is inserted into the knowledge model. The cases were inserted with their diagnosis, and were set as solved in the knowledge model. Then one case at the time was extracted from the knowledge model (set as not solved), and run through the CBR cycle against the knowledge model (containing all the other cases), for producing a new diagnosis. The found diagnosis was then logged and sent as output, before the case was introduced into the knowledge model again (with its correct diagnosis and set as solved).

A small client was made in Java using an ORB made with the Java IDL compiler from the IDL file. The client is used for calling test method in the server object, and start the testing of the knowledge model. The results of this can be found later in this chapter.

5.4.3 Dynamic Imager modules.

There were in all seven dynamic imager modules produced especially for our system. These modules' functionality has been described earlier in this chapter, and we will here describe some implementation specifics, and describe the input and output for each of these modules.

The "Split" module.

This module is used for getting the sequence for a specific slice of the breast. The module accepts as input the whole T1 image sequence, and as a user input, the number of an image in the whole sequence, that is inside the sequence wanted.

The "motion segmentation" module.

Accepts an image sequence as input and executes a motion segmentation algorithm with the innput sequence images. The module output the buildt motion image. In this module we do not only use the normal motion segmentation algorithm, as motion between the frames in an image sequence would make almost all points be countet. Instead, in this module, we calculates motion for a 3x3 neighboring area of eatch pixel. This seems to give quite a good result.

The "region growth" module.

The "region growth" module accepts as input an image, and a ROI. The module then utilizes the region growth algorithm on the input image to produce a new ROI. The module finds the pixel with the highest signal intensity inside the input ROI, and uses this as the "seed point" for region growth. This module also accepts a user input of a "cutoff" value. This value is the percentage for the cuttof point for the region growth. That is, for a pixel, determining if it can be included in the growing region, is determined by its signal intensity in comparison to the highest and lowest signal intensity of the input image, and the "cuttoff" percentage value.

The "periph" module.

The "periph" module takes the T1 slice image sequence, a ROI defining the whole tumor, and a ROI defining the core part of the tumor as input, and finds if there has been faster penetration of contrast agents in the peripheral parts of the tumor, than in the core. It does this by comparing the average signal values in the peripheral parts aganist the core parts, during the injection of the contrast agent. As the contrast agent penetrates the breast just before image 3, the assumption is that if the following holds true, the tumor has a higer peripheral penetration than in its core.

- The average values for pixels in the edge before injection is less than the average values in the edge after injection.
- The average values for the pixels in the core before injection is less than the average values in the edge before injection.

- The average values for the pixels in the core after injection is less than the average values in the edge after injection.

As we had only one complete test set, we make no claims to the validity of this assumption. A discussion on this assumption can be found in chapter 6.

The "T1 data" module.

The T1 data module has three main tasks. It produces a T1 signal increase graph, a T1 signal increase value, and it categorizes the T1 increase graph. It accepts as input an image sequence (with 9 images from one slice of a breast), and a ROI (for our network, the ROI from the "region growth" module). The signal increase graph is produced by averaging the signal values for the pixels inside the ROI, for each image. The graph therefore gets 9 values. The T1 signal increase value, is on the other hand calculated as the percentage difference between the second and the fourth image in the sequence, as these are from before and after the injection of the contrast agent. As for the categorization of the graph, an artificial neural network is used. The network has 10 input nodes (9 for the graph points and one with an innout of 1), 8 hidden nodes (also one with an innout of 1), and 5 output/resulting nodes (one for each of the five graph types). The output nodes that get the highest output define the type of an input graph. The input to the network is the 9 values from the graph, normalized to values between 0 and 1.

A special module was made for the backpropagation training of the weights in the network. 70 of the patient cases were used for training, and the training was stopped after 60 of the cases were correctly classified. A learning rate of 0.05 was used.

The T1 data module gives both the T1 percentage signal increase value, and the graph type as integer output. The graph type is then a value between 1 and 5, for the five graph types. It can also output the graph itself as a dataset, for displaying for manual insection.

The "T2* data" module.

The T2* data module accepts the T2* image sequence and a ROI as input (again the ROI is in our network defined by a region growth module). This module averages the signal values found inside the ROI for each of the 40 images in the sequence. It then finds the average signal intensity of the first 10 images (before contrast agent injection), and then calculates the difference between this value and the *lowest* signal value after the tenth image. The T2* signal percentage decrease value is the output value of this module, and again the graph itself can be sent as a dataset output for displaying for manuel inspection.

The "send data" module.

The send data accepts the output of the T1 data, T2* data and periph modules, as input. It also accepts, as user input, the menstrual status of the patient, and the spiculele status of the tumor in question. The user also has an "activate" button in this module, for sending the ouput. The output of this module is the string of finding and values that the corbacreek module accepts as input. When pressing the "activate" button, the user sends the output, and if the module is connected with a corbacreek module, will therefore activate that module.

The "corbacreek" module.

The corbacreek module accepts as input a string of semicolon separated relation types, and their values. It also accpets a file-path as input. The filepath is used to the CORBA JavaCreek server object to try to load a knowledge model. As user input it accepts a case-name and description for the case. The module that connects to the JavaCreek CORBA server object and starts calling the methodes for loading the specified knowledge model, bulding the case, and activating the RETRIEVE and REUSE parts of the CBR cycle with that case. After these parts of the CBR cycle, the modules display the results of the cycle, and asks the user if the case should be incorperated in the knowledge model. This can be seen as the REVICE part of the CBR cycle, as JavaCreek in itself has no special methodes for this part of the cycle. If the user selects to accept, the RETAIN part of the CBR cycle is activated. If the user selects not to accept, the module terminates and the RETAIN part of the CBR cycle is not executed on the server with the present case.

5.5 Example running and Results.

5.5.1 Introduction

In this section we will describe a couple of test run of our system, and present the results of these test. One of the tests was made with a small Java client that activated the "testFocus" method in the JavaCreek server object, for leave one out cross validation of the knowledge model, while the other test was of the whole system, including both the JavaCreek server and the Dynamic Imager network. This last test was with one image data set acquired from RIT especially for this.

5.5.2 The Java client.

The Java client that was built especially for activating the testFocus() method in the JavaCreek CORBA server object was used for this test. When activated, the testFocus method loaded the knowledge model, and populated it with all our patient cases. Then a "leave one out cross-validation" test was run with these cases. One case at the time was extracted from the knowledge model, and the CBR cycle was activated with the case. The result was logged, before the case was inserted into the knowledge model again (with its correct diagnosis). There were 127 cases in total, with 70 diagnosed as malignant, and 57 as benign. This gave us the following results:
True positive: 55
True negative: 50

As the JavaCreek correctly diagnosed 55 of the 70 malignant cases, the JavaCreek system with our knowledge model, gets a sensitivity of 79 percentage. The system correctly diagnosed 50 of the 57 benign cases; which gives a specificity of 89 percentage.

5.5.3 Dynamic Imager client.

Here we will present a test of the Dynamic Imager client, and its CORBA connection to the JavaCreek CORBA server object. This test was run with one image dataset provided by RIT, for just this reason. The dataset consists of the T1 and the T2* image sequences for one patient case; which was diagnosed having a malignant tumor.

Before using the Dynamic Imager module network we had produced for this task, we first had to start a CORBA naming service (for handling the IOR for the server object), and then the JavaCreek CORBA server object. The server object then binds its IOR to the naming service, and is ready for use.

Loading, displaying and defining the tumor.

After loading the two image sequences for T1 and T2*, we had to find manually the tumor in the T1 display module. The image number of an image where the module was apparent was then inserted into the "split" module. Then we drew a figure covering the tumor in the displayed images of both the T1 and T2* data set. The displayed images of the breast and the drawn figures for both the T1 and T2* data can be seen in image 5.10. The results of the "motion segmentation" modules for T1 and T2* can be seen in image 5.11

The motion segmentation module together with the user defined ROI is used for finding the final ROI in the "region growth" module. In figure 5.12, we can see the result of the region growth modules in our network.

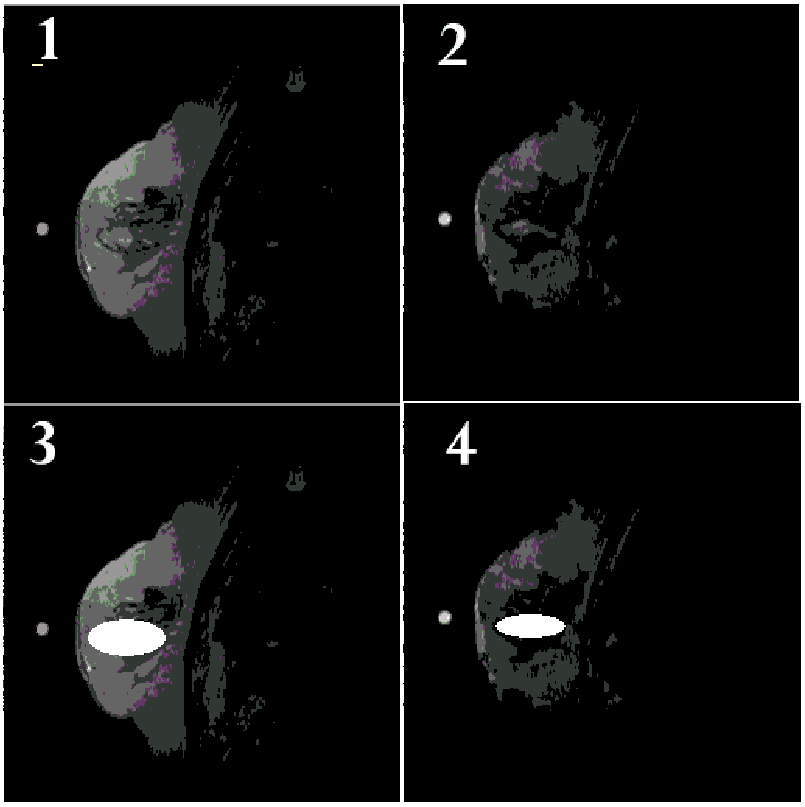

Figure 5.10: 1: T1 data image of tumor, 2: T2* data image of tumor, 3: T1 with user ROI, 4: T2* with user ROI.

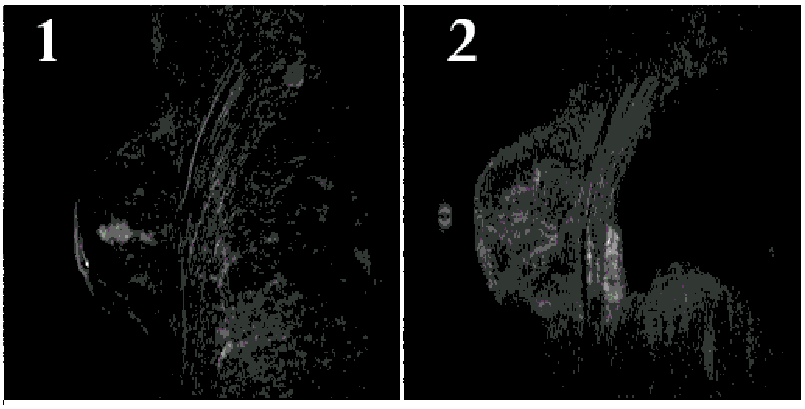

Figure 5.11: 1: T1 motion segmented, 2: T2* motion segmented.

Figure 5.12: Different T1 ROI's made from the motion segmentation image, and the user defined T1 ROI

The data found.

Using the ROI found with one of the "region growth" modules, and the T1 sequence for the relevant slice of the breast, the "T1 data" module can calculate the signal increase graph, the graph type and the signal increase percentage. The graph found can be seen in figure 5.13. This graph was categorized as type 4, and the T1 signal increase percentage was found to be 189%. The actual graph type was 5 and the actual signal increase percentage found by the scientist at RIT, was 203%. As the found walue was above 90%, this was translated to a "highenhance" finding in the pushdata module.

The module for finding high peripheral penetration of contrast agent, found a high penetration, which corresponds with the manual finding for this case.

The "T2* data" module, used for finding the percentage signal intensity drop, found a signal drop on 25% in comparison to 23% which was the RIT scientist's finding. The graph for the T2* signal intensity can also be seen in 5.13. As the found value was above 20%, this was translated to "highdrop" in the pushdata module.

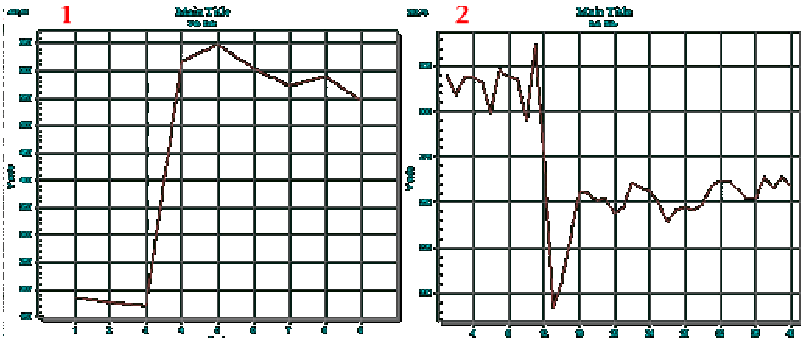

Figure 5.13: 1: T1 signal intensity graph. 2: T2 signal intensity graph.

Activating the CBR cycle in JavaCreek.

After all the data was found, and the menstrual status and spiculele finding were inserted, the "corbacreek" module could be activated. The CBR cycle in JavaCreek was then activated with the case, and a diagnosis was produced. The diagnosis found by the system was that the found tumor was malignant, as was the manual diagnosis from the scientist at RIT. An image of the Dynamic Imager software displaying the diagnosis, and asking if the user wants to incorporate this into the knowledge model can be seen in figure 5.14. As seen in the figure, the user is first presented with the best case from the RETRIEVE part of the CBR cycle. As seen, all the strengths are 0.5, from the fact that explanation strengths are still not supported in

JavaCreek. The next information is the found diagnosis, with a short quote on how sure the server is that the diagnosis was correct. Together with this quote, the system should actually print out an explanation for why the diagnosis was as sure as the quote stated, but the part of JavaCreek producing this explanation is also still under development.

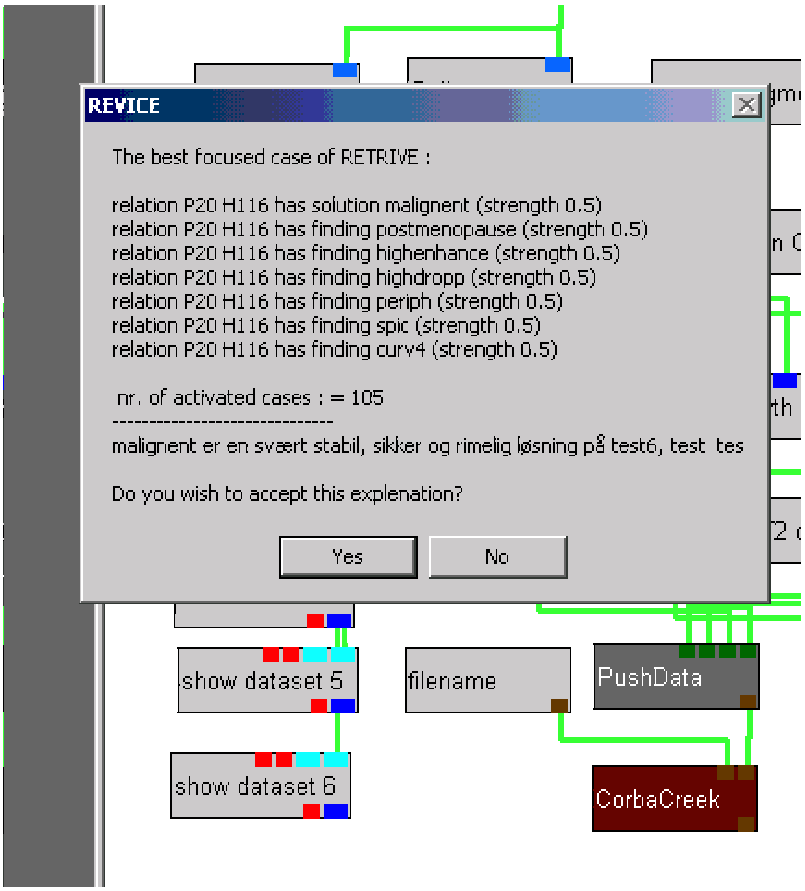

Figure 5.14: Displaying diagnosis from JavaCreek.

Chapter 6

Result evaluation

6.1 Introduction.

In this chapter we will evaluate our proposed system solution to the goal definition as described in chapter 1, and the results from our prototype system agenst other results in the domain of diagnosing malignancy in MR image sequences. As much of our work is based on the work presented in [6] [8] [9] [12], there are several different sources for data to compare our results with.

In our goal definition we divided our goal into three sub goals. We will utilize these three sub goals as basis for determining the degree of success in solving our main goal. The sub goals were:

- Connecting the Dynamic Imager system with the JavaCreek CBR system.
- Image segmentation and classification with Dynamic Imager, to produce data for JavaCreek.
- Diagnosis of malignancy with the JavaCreek CBR system.

In this chapter we will first discuss the proposed solutions for each sub goal. For the sub goal of diagnosis, we will compare our results with other relevant works. For the sub goals of connecting the JavaCreek and Dynamic Imager software, and the Dynamic Imager segmentation and classification, the evaluation will be based on our prototype implementation, and the results from trying our system with the provided dataset.

6.2 Malignancy diagnosis and the knowledge model.

6.2.1 Introduction

Here we will take a closer look at the results we got from our test of the JavaCreek CBR system with our proposed knowledge model (as described in chapter 5). For the test we used 127 patient cases, where 70 were malignant. The results we got were:

- True positive: 55 (the number of malignant tumors detected correctly).
- True negative: 50 (the number of benign tumors detected correctly).
- Sensitivity: 79% (55 malignant tumors of 70 possible detected, gives a success rate of 79% in detecting malignancy correctly).

Parameters	True positive	True Negative	Sensitivity	Specificity
Min.enhanc.	59	52	84	91
Fisher's linear.	56	53	80	93
Prob. Neural Net	58	51	83	89
Backprop. Net	58	47	83	82
CC	56	45	80	79

Table 6.1: Classification after noise reduction of data set.

Parameters	True positive	True Negative	Sensitivity	Specificity
Min.enhanc.	58	51	83	89
Fisher's linear.	56	51	80	89
Prob. Neural Net	57	49	81	86
Backprop. Net	55.5	45.5	79	79
CC	52	46	74	81

Table 6.2: Classification before noise reduction of data set.

- Specificity: 88% (50 benign tumors of 57 possible detected, gives a success rate of 88% in detecting benign tumors correctly).

These results will form the basis for our evaluation together with results from other works that utilize the same data set used in our works, or parts of it.

6.2.2 Results from others.

The patient dataset used for our test of our proposed JavaCreek knowledge model, was the same as used in several other articles. Among other, in Srmo [12]. The main results from the different methods tried with the T2* data in the article (as described in chapter 3.4, can be seen in table 6.1 and table 6.2.

The patient dataset was also used as basis for the article "Breast Lesions: Evaluation with Dynamic-enhanced T1-weigted MR Imaging and with T2*-weighted First-Pass Perfusion MR Imaging" [8], as described in chapter 3.4. The scientist using both the T1 and T2* data from the data set, got by semi-automatic diagnosis the results for different methods described in table 6.3

6.2.3 Result explanation and specification.

From the results of other methods using the same data set, we see that the use of the JavaCreek software with our knowledge model yields almost, but not quite, as good results as using the methods for utilizing the T2* data as seen in table 6.1 and table 6.2. When comparing with the results from using

Parameters	True positive	True Negative	Sensitivity	Specificity
T2* signal decr.above 20%	57	54	79	93
T1 grafhs.	63	46	88	79
T1 signal incr. above 90%	64	39	89	67
Spiculele	60	46	83	79
Peripheral	53	46	74	82

Table 6.3: Results from different methods presented in [8].

only single methods as described in table 6.3. The reasons that our proposed method does not compare to the T2* based methods, can be several, but we will here present two.

T1 data vs. T2* data.

In our knowledge model we utilize findings from both the T1 and the T2* data set as basis for the diagnosis. However, in "Differentiating benign and malignant breast lesions with T2*-weighted first pass perfusion" [12], and in [8], parts of the conclusions and discussions states that the use of the T1 data for locating a tumor, and then using this to acquire T2* data of the tumor for the diagnosis, is better than any use of the T1 data set for diagnosing. This may indicate that the use of the findings from the T1 data sets may actually diffuse the diagnosis process, in comparison to using only the T2* data. This may be seen by comparing the results in table 6.1 and 6.2 against table 6.3

The use of relation strengths, or not?

In our knowledge model we did calculate and include strengths for all relations. This was not explicitly mentioned and explained in our description of our system. The reason for this was simply the fact that the JavaCreek CBR software does not yet support the use of the relation strengths part of the CREEK CBR model. This, among other things, leads to the fact that the relevance of the different findings was not weighted. Among other things, the T2* signal decrease was far stronger weighted than any of the findings based on the T1 data set, or the observable findings. Also the findings from the T1 data were found to have different weights for the diagnosis. For instance, if the tumor had a spiculele shape or a high peripheral penetration of contrast agent, it was found to have not as strong influence towards a diagnosis of malignancy as the T1 signal increase percentage or curve type. The fact that the JavaCreek did not support the use of weighing, did unfortunately not allow us to test the influence of the weights on the diagnosis, or the correctness of the calculated weights. The weights values found by normal statistical calculations, can be seen in the "mrcancerdomain.java"

file, in appendix A.

6.3 The Dynamic Imager client.

6.3.1 Introduction.

As only one full patient case image data set was provided for this work, the evaluation of the different Dynamic Imager modules was hard. However, this does not imply that our approach for solving the main problem can't be evaluated. In this section we will take a closer look at the image processing methods implemented with the modules, both the segmentation and the classification part. We will also evaluate the use of CORBA for connecting the Dynamic Imager and the JavaCreek software.

6.3.2 The segmentation process.

The segmentation process for both the T1 and T2* image data, are both loaded by the "Import Binary" module, a module provided in the Dynamic Imager software. The "show dataset" module (and the "orthoslicer 3D" module for the T1 data set) that is used for the manual ROI definition, is also a module provided with the software. However, the rest of the modules used in the segmentation process are produced especially for the task at hand.

The segmentation process is in implementation handled in a divided fashion. First motion segmentation is used to make a tumor more clear and easier to see both manually and for the system. The result of the motion segmentation of both the T1 and the T2* data can be seen in image 5.11.

In the motion segmentation image from the T1 data set, spotting the tumor is quite easy. However, in the T2* motion segmented image, spotting the tumor is not as intuitive as in the T1 motion segmented image. As the position of the tumor in both segmented images will (at least to some degree) correspond, this is not a problem, as the second part of the segmentation process is based on manual ROI definition as well as the motion segmented images.

In the second part of the segmentation (for both the T1 and the T2* data), the final ROI is calculated from the motion segmentation produced image, and a manually defined ROI, by utilizing region growth on the result of the motion segmentation modules, inside the manually defined ROI's (the original image and the manual defined ROI's are showed in image 5.10). As the user have access to both the original image and the motion segmentation produced images when defining the ROI's, he can utilize the T1 data as a reference when defining the ROI for the T2* data. Examples of final ROI's defined by the region growth module, can be seen in image 5.12.

As only one data set was available for the implementation and testing of our prototype, we can in no way state that the methods used are sufficient for the task at hand. However, we will state that the methods may at least be usable for a domain expert, as a basis for further development of the concept of the overall system design as described in chapter 5.3.

6.3.3 The classification process.

The modules used for classification in our prototype implementation all use either the T1 data sequence of the selected slice, or the T2* data sequence, and one or more defining ROI's. The different modules and their input are shown in figure 5.6, figure 5.7 and figure 5.8. As seen in the figure, the peripheral module needs ROI's defined by two "region growth" modules. We found cutoff values for these modules by trial and error. We implemented the process of selecting a cutoff-value in the region growth module was done by using a slider for selecting values, where the resulting ROI changed as the user used the slider. This was necessary as we did not have enough data sets to check if any global values could be found. This also holds true for the region growth modules used for the modules specified for gathering the T1 and the T2* specific data (the "T1 data" and the "T2 data" modules).

In the T1 data module we had an artificial neural network for categorizing the T1 intensity signal increase. This neural network was described in chapter 5.4.3, and normal backpropagation was used for the training of the network. As the graph values for each patient case were provided, we could train this network. We selected 70 graphs from all the five graph categories (14 graphs from each categories). As the graphs provided were not produced in our prototype system, and that it was the shape of the graph and not the values of its data that was the selecting factor for the categorizing of the cure, we normalized the data to values between 0 and 1. As the graph found and used was highly determined by the ROI used, this was done for ensuring that the network would produce results that could be used with our produced graphs. However, as the network was used for categorizing the graph produced for the one patient image set provided, the result indicated that more and better-tuned training might be in order. The graph produced should by all rules be categorized as a type 5 graph (as seen in image 5.13, but the resulting categorizing done by the artificial neural network yielded a type 4 graph for our test patient case. This, together with the fact that we stopped training after only 60 of the 70 patient graphs were correctly classified, indicates that this network can be greatly improved.

For the T1 signal increase percentage, the T2* signal decrease percentage and the determination of peripheral penetration, as described in chapter 5.4.3, the Dynamic Imager modules did produce results that corresponded well with the results earlier for the specific data set with other methods. However, as our approach for acquiring the results differ from other methods

used (especially, our ROI's does of course not correspond with earlier tests), some differences are to be expected. These differences make a comparison of the results from our methods and earlier found results difficult, as a comparison by values would not yield any valid results.

6.3.4 The use of CORBA for connecting CBR and image processing software.

The CORBA interface to JavaCreek was tested using both the Java client used for the testing of the knowledge model, and the Dynamic Imager client module. All tests and use of the CORBA server extension for JavaCreek worked as intended, as long as the use was in compliance with the intended use. We did not manage to get the use of CORBA in itself to trigger an error, or fail in achieving the intended goals. However, using the interface wrongly was fully possible, and would result in errors. An example of such use would be to call the methods in the server object in the incorrect order.

The use of the CORBA client API, enabled us to easily produce a Dynamic Imager module for connecting to JavaCreek, controlling the CBR cycle and presenting the resulting diagnosis. As the API we produced for this was general, this gives the possibility for using this approach for utilizing JavaCreek for other problems. In Dynamic Imager use of this method would imply producing a new module using the CORBA server API. As the CORBA standard used is supported by several computer languages and compilers, this approach would also be quite usable for connecting other software, or producing standalone clients.

6.4 Result evaluation.

We will now again revisit the sub goals presented in chapter 1, and compare the goals with our results to evaluate our work. This will give an indication for our overall success.

- Connecting the Dynamic Imager system with the JavaCreek CBR system: This part goal must be said to be achieved with good results. Not only did we manage to connect the Dynamic Imager with the JavaCreek system, but also gave an API for connecting to JavaCreek with any application that can be produced or enhanced to support CORBA.
- Image segmentation and classification with Dynamic Imager, to produce data for JavaCreek: The success of this sub goal can be questioned, as most of the results could not be evaluated by the lack of test data. However, we assumes that the module network produced can be easily used by an domain expert to produce good results, as there is constant presentation of any findings, and there are ample possibilities for "tweaking" of the modules.

- Diagnosis of malignancy with the JavaCreek CBR system:
  The results for the diagnosis procedure using the JavaCreek knowledge model presented in this thesis, was presented earlier in this chapter. The results were comparable or better than most T1 data based methods presented earlier in this thesis. However, the system did not impress when compared to the results from the newer T2* based results presented in chapter 4.3. However, his did no surprise as our approach was mostly influenced by the T1 based methods, and that the JavaCreek system does not yet support the use of explanation strengths.

In our goal description, the handling of time in the data set was especially mentioned. As the JavaCreek system in itself does not support cases over time as of today, we worked from the assumption that handling the time aspect was best done before the JavaCreek was utilized for the diagnosis. The T1 signal intensity graphs categorizing, was a developed method for translate time based knowledge to symbolic values in this knowledge model, and shown to be of use in diagnosing malignancy. The use of motion segmentation to define the ROI's needed, was on the other hand based our assumption that this would utilize the extra information the time based data could provide, compared to only using static images. In our opinion, based on the achieved results, we still assumes this to hold true, and that this method can be further developed toward a fully automatic ROI definition tool for this knowledge domain, as soon and if, more image patient data sets are made available.

From the evaluation of the sub goals, and the handling of time in the data sets, we can conclude that our mayor goal of producing a system that by connecting existing software for image processing and artificial intelligence witch could be used for diagnosing malignancy in breast tumors from MR data was mostly successful. Although parts of our system did not give results that were better than other automatic or semiautomatic methods used for the same diagnosing, we think that the overall system design, as a whole, can be seen to solve the main goal.

Chapter 7

Summary and future work.

7.1 Summary.

Before presenting the areas where we feel that future work from this thesis may yield worthwhile results, we will present a summary of the thesis.

In chapter 1 we presented our goal for this thesis. The main goal was to connect the Dynamic Imager image-processing environment with the JavaCreek artificial intelligence system, for approaching an automatic system for diagnosis of malignancy in breast MR images. The image data sets was taken over time as contrast agent was injected into the patient, and contained data gathered using both the T1 and T2* MR techniques. This meant that using existing, and developing new methods for utilizing the time dependent information in the data set, was a great opportunity to increase the diagnosis accuracy, in comparison with methods using only static data.

As the goal of our work included testing methods for diagnosis malignancy in tumors from MR data sets, we early (in chapter 5 stated that a prototype system had to be produced and used for the testing. This prototype was based on the results from earlier work in diagnosing malignancy in breast MR images, and from the field of image processing. Dynamic Imager and JavaCreek was presented for explaining the selection of this systems for the task at hand, and we also evaluated two computing network standards, the TCP/IP client/server approach and the CORBA approach, for connecting the JavaCreek and the Dynamic Imager software together. Earlier results, software presentation, image processing methods, the MR data gathering methods and the standards for network computing, was all presented in chapter 3 and chapter 4.

In chapter 5, we first presented our reasons for selecting the CORBA standard for the integration of Dynamic Imager and JavaCreek. The reasons for selecting CORBA was several, but the main reasons where the ease of implementation, and the added possibilities compared to the TCP/IP client/server based approach. From the selection of CORBA, we went on to describe our overall design strategy for our prototype system, from the use of Dynamic Imager modules for image processing, to utilizing the CORBA as an interface to JavaCreek, and finally to utilizing the JavaCreek CBR system for the diagnosing.

Next we described the knowledge model that we produced for JavaCreek for the knowledge domain, based on the earlier works in the filed presented in chapter 3 and 4. We also presented the work done for utilizing CORBA for the integration process, the Dynamic Imager modules produced, and the module network used in Dynamic Imager for the image processing part of the overall system. Finally, in chapter 5, we presented the running of the overall system, and the results acquired.

In chapter 6 we compared the results we of our system, with comparable system and methods. From this comparison we concluded that although our goal was in most areas and for the system as a whole successful, much work

still remains on parts of the system.

7.2 Future work.

As stated in the last section, much work still remains for our system. Also, several areas has during the work on this thesis, shown itself to be worthy of closer inspection and more work. We will at the end of this thesis now present these areas that we find of most importance.

7.2.1 General knowledge in JavaCreek.

The produced knowledge model for the knowledge domain included general domain knowledge as strengths to the implications in the model. These strengths should influence the reasoning process in JavaCreek. Unfortuanly, the JavaCreek system does not fully support this part of the Creek CBR method, and therefore the results obtained by using this knowledge model might be improved a great deal when the use of explanation strengths become supported. As the JavaCreek system is under constant development, this part will probably be available in the not too far future, and should then be reevaluated.

7.2.2 The artificial neural network.

In our system approach, we used a module in the Dynamic Imager part of the system; to categorize the T1 based signal intensity graphs. At NTNU there is now work being done to include use of artificial neural networks directly into JavaCreek, for similar translation from sub symbolic knowledge, to symbolic knowledge that can be used in the reasoning process. As our neural network showed sign of not being trained sufficient, the use of such an integrated network would be of great interest, as a connection of the reasoning mechanism in JavaCreek and the training of such an integrated network has been mentioned. Such integration would also imply that the learning of the network could actually change as more cases were incorporated into the knowledge model, and more knowledge was made available.

7.2.3 The use of motion segmentation for automatic ROI definition.

In the Dynamic Imager part of our system, we utilized motion segmentation in the semiautomatic process of defining ROI's. This method could well be made fully automatic if more work was done on this field, and the method was connected with other image segmentation methods. An requirement for further study into this would of course be that more test image data sets was available.

7.2.4 The JavaCreek CORBA server.

Finally, we will mention the JavaCreek CORBA server. As of today this server can be used for utilizing JavaCreek from any existing or new systems, as long as they can utilize CORBA. However, as of today, a serious shortcoming exists in the server. Only one user at a time can access the server, and must fully finish the CBR cycle with a case before another user may use the server. The users is not warned if the server is already in use, and this can therefore lead to errors and even crashes. Further development of the CORBA server could remove this shortcoming, and make a knowledge model usable for several simultaneous users, making any knowledge model a true knowledge repository, accessible by users around the world.

Bibliography

[1] Agnar Aamodt : *A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning* PhD thesis at University in Trondheim, NTH, 1991, Trondheim.

[2] Agnar Aamodt : *Explenation-Driven Retrival, Reuse and Learning of Cases* Proceeding of EWCBR-93, First European Workshop on Case-Based Reasoning, Kaiserlautern, Nov, 1993. pp 279-384.

[3] Agnar Aamodt : *Explenation-driven case-based reasoning* In S. Wess, K. Althoff, M. Richter (eds.): Topics in Case-based reasoning. Springer Verlag, 1994. P. 274-288

[4] Agnar Aamod, Enric Plaza : *Case-Based reasoning; Foundation issues, methodological variations, and system approaches.* AI Communications, Vol.7, No. 1, Marc 1994, pp. 39-59

[5] Ceetron : *www.ceetron.com*

[6] Kjell Arne Kvistad : *MR in breast cancer*, A climical study Dr. theisis, Norwegian University of Science and Technology, Faculty of Medicine, Trondheim - Norway. ISBN 82-7964-015-0.

[7] K. A. Kvistad, S. Lundgren, H. A. Fjsne, E. Smenes, H.-B Smethurst, O. Haraldseth. *Differenting benign and malignant breast lesions with T2*-weighted first pass perfusion imaging* Acta Radiologica 40 (1999) : 45-51. ISSN 0284-1851.

[8] Kjell A. Kvistad, Jana Rydland, Jari Vainio, Hanne B. Smethurst, Steiner Lundgren, Hans E. Fjsne, Olav Haraldseth. *Breast Lesions: Evaluation with Dynamic Contrast-enhanced T1-weighted MR Imaging and with T2*-weighted First-Pass Perfusion MR Imaging* In Radiology 2000; 216:545-553

[9] Kjell A. Kvistad, Inger J. Bakken, Ingrid S. Gribbestad, Benny Ehrnholm, Steinar Lundgren, Hans E. Fjsne, Olav Haraldseth. *Characterization of Neoplastic and Normal Human Breast Tissues With In*

Vivo HMR Spectroscopy in Journal of Magnetic resonance imageing 10: 159-164.

[10] CORBA&S, Object Management Group (OMG) : *www.omg.com*

[11] Marvin Minsky : *A Framework for Representing Knowledge*, In P. Winston (ed.): The psychology of computer vision. McGraw-Hill, 1975, p. 211-277

[12] Frode Søermo : *Plausible Inheritance*, Semantic Network Inference for Case-Based Reasoning. Master thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, 2000

[13] Jain, R. : *Segmentation of Frame Sequences Obtained by Moving Observer.* Report GMR-4247, General Motors Research Labaratories, Warren, Mich.

[14] Muerle, J.L, and Allen, D.C [1968]. *Experimental Evaluation of Techniques for Automatic Segmentation of Objects in a Complex Scene.* In Pictoral Pattern Recognition, (G.C. Cheng et al., eds.) Thompson, Washington, D.C.

[15] Brice, C.R., and Fennema, C.L. [1970]. *Scene Analysis Using Regions.* Artificial Intelligence, vol. 1, pp.205-226.

[16] McCulloch, W.S, and Pitts, W.H [1943]. *A Logical Calculus of the Ideas Imminent in Nervous Activity.* Bulletin of Mathematical Biophysics, Vol. 5, pp 115-133.

[17] Geir Torheim, Fred Godtlibsen, David Axelson, Kjell Arne Kvistad, Olav Haraldset, Peter A. Rinck. *Feature extraction and classification of dynamic contrast-enhanced T2*-weighter breast image data* Department of Anesthesia and Medical Imaging. Norwegian University of Science and Technology. Trondheim, Norway. Under develpment, part of the work presented at ISMRM 2000

Appendix A

The Knowledge Model

```
package corbacreek;
import javacreek.representation.*;
import java.io.*;
import java.util.Vector.*;
import javacreek.*;
import java.util.StringTokenizer;

public class mrcancerdomaintest {
  public mrcancerdomaintest(KnowledgeModel km){
    km.name = "MrCancerDomain";
    km.title = "MR - Breast Cancer domain by Jo Skjermo.";
    km.comment ="This domain is used to demonstrate the corbacreek interface.";

    //public String casen()=new String();
    System.out.println("Adding a few entities...");

    try{
     //The parameters in the domain
     Parameter topNode = new Parameter(km,"top node","Top node of the parameter tree.",null);
     Parameter observable = new Parameter(km,"observable","Observable Parameter.",null);
     Parameter measurable = new Parameter(km,"measurable","Measurable Parameter.",null);
     Parameter diagnosis = new Parameter(km,"diagnosis","The diagnoses for this domain",null);

     //The observable Parameters
     //Parameter age = new Parameter(km,"age","The patients age",null);
     Parameter mens = new Parameter(km,"mens","Menstruational status",null);

     //The measurable Parameters
     Parameter si_curv = new Parameter(km,"si_curve","Signal curve of T1-test",null);
     Parameter tissue = new Parameter(km,"tissue","Peripheral penetration of outer tissue of tu
     Parameter shape = new Parameter(km,"shape","The shape of the tumor",null);
     Parameter t1 = new Parameter(km,"t1","Signal enhancment in tumor with T1",null);
     Parameter t2 = new Parameter(km,"t2","Signal decrease in tumor with T2",null);
```

```
//The Solutions/diagnoses
Finding malignent = new Finding(km,"malignent","The tumor is malignent",null);
Finding benign = new Finding(km,"benign","The tumor is benign",null);
Finding notknown = new Finding(km,"notknown","The tumor can be anything",null);

//The possible age Findings.
//Finding lowage = new Finding(km,"lowage","The patient is below 44years old",null);
//Finding highage = new Finding(km,"highage","the patient is 44 years or older",null);

//The possible mens Findings.
Finding premenopause = new Finding(km,"premenopause","The patient is premonopause",null);
Finding postmenopause = new Finding(km,"postmenopause","The patient is postmenopause",null
Finding hrt = new Finding(km,"hrt","The patient is undergoing hormone replacement therapy'

//The possible si_curv findings.
Finding curv1 = new Finding(km,"curv1","T1 time-signal intensity curve classification val
Finding curv2 = new Finding(km,"curv2","T1 time-signal intensity curve classification val
Finding curv3 = new Finding(km,"curv3","T1 time-signal intensity curve classification val
Finding curv4 = new Finding(km,"curv4","T1 time-signal intensity curve classification val
Finding curv5 = new Finding(km,"curv5","T1 time-signal intensity curve classification val

//The possible tissue finding
Finding  periph= new Finding(km,"periph","Peripheral penetration noticed",null);
Finding  noperiph= new Finding(km,"noperiph","No periheral penetration spotted.",null);

//The possible shape finding
Finding  spic= new Finding(km,"spic","The tumor has a piculele shape",null);
Finding  nospic= new Finding(km,"nospic","The tumor has a normal shape",null);

//The possible t1 finding
Finding  lowenhance= new Finding(km,"lowenhance","The T1-enhancing is below 90%",null);
Finding  highenhance= new Finding(km,"highenhance","The T1-enhacing is above or equ 90% ",

//The possible t2 finding
Finding  lowdropp= new Finding(km,"lowdropp","The T2-decrease is below 20%",null);
Finding  highdropp = new Finding(km,"highdropp","The T2-decrese is above or equ 20%",null)

//The Subclasses of topNode
new SubclassOf(observable,topNode);
new SubclassOf(measurable,topNode);
new SubclassOf(diagnosis,topNode);

//The Subclasses of the observable Parameters
//new SubclassOf(age,observable);
new SubclassOf(mens,observable);

//The Subclasses of the measurable Parameters
new SubclassOf(si_curv,measurable);
new SubclassOf(tissue,measurable);
```

```
new SubclassOf(shape,measurable);
new SubclassOf(t1, measurable);
new SubclassOf(t2,measurable);

//The possible Soulutions (subclasses and solution)
new HasValue(diagnosis, malignent);
new HasValue(diagnosis, benign);
new HasValue(diagnosis, notknown);

//Possible findings for the observable parameters

//the possible findings for mens.
new HasValue(mens,premenopause);
new HasValue(mens,postmenopause);
new HasValue(mens,hrt);

//Possible findings for the measurable parameters
new HasValue(si_curv,curv1);
new HasValue(si_curv,curv2);
new HasValue(si_curv,curv3);
new HasValue(si_curv,curv4);
new HasValue(si_curv,curv5);

new HasValue(tissue,periph);
new HasValue(tissue,noperiph);

new HasValue(shape,spic);
new HasValue(shape,nospic);

new HasValue(t1,lowenhance);
new HasValue(t1,highenhance);

new HasValue(t2,lowdropp);
new HasValue(t2,highdropp);

//Some implications
//Relation Relasjon = new Relation();

//Implication between age and mens?



Implies Imp = new Implies(curv1,benign);
Imp.setExplanationStrength(0.9);
Implies Imp1 = new Implies(curv2,benign);
Imp1.setExplanationStrength(0.8);
Implies Imp2 = new Implies(curv3,malignent);
Imp2.setExplanationStrength(0.6);
Implies Imp3 = new Implies(curv4,malignent);
```

```
 Imp3.setExplanationStrength(0.75);
 Implies Imp4 = new Implies(curv5,malignent);
 Imp4.setExplanationStrength(0.9);
 Implies Imp7 = new Implies(noperiph,benign);
 Imp7.setExplanationStrength(0.8);
 Implies Imp8 = new Implies(periph,malignent);
 Imp8.setExplanationStrength(0.75);
 Implies Imp9 = new Implies(spic,malignent);
 Imp9.setExplanationStrength(0.8);
 Implies Imp10 = new Implies(nospic,benign);
 Imp10.setExplanationStrength(0.85);
 Implies Imp11 = new Implies(lowenhance,benign);
 Imp11.setExplanationStrength(0.6);
 Implies Imp12 = new Implies(highenhance,malignent);
 Imp12.setExplanationStrength(0.6);
 Implies Imp13 = new Implies(lowdropp,benign);
 Imp13.setExplanationStrength(0.7);
 Implies Imp14 = new Implies(highdropp,malignent);
 Imp14.setExplanationStrength(0.7);

 Implies Imp15 = new Implies(premenopause,benign);
 Imp15.setExplanationStrength(0.2);

 Implies Imp16 = new Implies(postmenopause,malignent);
 Imp16.setExplanationStrength(0.2);

 Implies Imp17 = new Implies(hrt,malignent);
 Imp17.setExplanationStrength(0.15);

 //Some eksamples of causes
 //new Causes(banana,monkeyFeeling);


//cases
Case case1=new Case(km,"examplecase1","benign",null);
//new HasFinding(case1,lowage);
new HasFinding(case1,curv2);
new HasFinding(case1,nospic);
new HasFinding(case1,noperiph);
new HasFinding(case1,lowenhance);
new HasFinding(case1,lowdropp);
new HasFinding(case1,premenopause);
new HasSolution(case1, benign);
case1.setStatus(Case.SOLVEDCASE);

//THE REAL CASES SHOULD COME HERE.
//HOWEVER, AS WE DID NOT GATHER AUTHORISATION
//TO REPRINT THEM HERE, THE CASES HAS BEEN
//REMOVED.
```

```
     }catch (ConceptAlreadyExistException e){System.out.println("ConseptAlreadyExists: "+e.getM
  }

  public static void main (String[] args)
  {
      KnowledgeModel km = new KnowledgeModel();
      new mrcancerdomaintest(km);
  }
}
```