University of Twente

Masters Thesis

Workflow Usage in the Healthcare Environment

Author:
Tristan Pothoven

Supervisors:
Dr. E.M.A.G. van Dijk
Dr. D.K.J. Heylen
Ir. J. G. Dannenberg
Ing. P. Noltes

In Association With

Topicus Care

April 22, 2010

Abstract

This research is about the application of workflow technology to improve the exchange of information concerning the delivery of care in residential homes. The healthcare domain is introduced and a general idea on the tasks performed there is provided. In general these tasks can be divided into two categories, i.e. that of administrative tasks and care-providing tasks. Care-providing tasks are directly connected to the delivery of healthcare to a client. These tasks are generally very divers and the actual execution differs from specialist to specialist. Administrative tasks are more related to the financial part of a client's situation, like insurance information used for billing. These tasks are generally static, meaning they rarely change. It is therefore more logical to try to apply workflow technology to the field of administrative tasks. This thesis includes a complete example of one of the possible paths that a client, from an administrative point, may traverse. This path starts at the moment the clients becomes in need of healthcare and ends at the moment the client does not require healthcare anymore.

A basic introduction to the usage of workflows and how workflow systems are designed is provided. An overview of the applications of workflows and the determination when workflows are best suited are provided to allow a determination of where workflow technology could be applied within the healthcare domain. The YAWL workflow system was chosen for its academic background and because it provides a near perfect match for the requirements we wished to see in a workflow system within our chosen domain.

During this research two different prototypes were designed; only the first one, the Topicus Task Manager, was implemented. The Topicus Task Manager, a system that tracks predefined tasks during their execution in a healthcare information system, was developed with the focus on providing the best guiding support while executing the tasks. The second prototype, the Snippet Factory, focuses on the ad-hoc creation of workflow models to allow an additional form of flexibility which workflows generally do not possess.

To test the prototype three different evaluation sessions with employees at Topicus were organized. During these three sessions the attention was focused on the functionality (analyst evaluation), the technical potential (developer evaluation) and a combination of both. The last evaluation session was held to obtain the best overall view and future potentials. The most important result was that function-wise the prototype provided a lot of potential and interesting possibilities for future use. There were however some concerns with the actual "look and feel" of the prototype. It was generally agreed however, that these flaws could easily be rectified.

The conclusion is that there are a lot of interesting points with regards to workflow application in the residential care domain. The most gain, as was expected, can be obtained by providing workflow support in the administrative field, where tasks are often repeated, have to be done in a predefined way and seldom change.

Contents

Chapter 1

Introduction

One of the common goals of all computer scientists is to automate processes that are clearly meant for automation (e.g. repetitive tasks) and facilitate human beings in their daily routines.

For this research, the focus lies within the healthcare domain. The amount of healthcare required in the Netherlands is growing rapidly. According to the Dutch central bureau of statistics (*CBS*) [31] the amount of people age 65 and above[1] is around 15% of the Dutch population (2,473,363 people). This number is expected to grow in the coming years to nearly 20% in ten years. This rate of growth is expected to continue till beyond 2040.

These numbers clearly indicate that the healthcare domain is one of the fastest growing branches of employment in the Netherlands. At the moment of writing (2010), healthcare organizations are struggling to keep up with their work. This is partially due to a substantial quantity of administrative tasks that comes with providing healthcare. These range from requests for government funding to creating client dossiers and updating them after a healthcare professional has seen or treated the client. At this moment, most of these administrative duties are performed manually by healthcare professionals or administrative employees.

So, the tasks in healthcare come in two different categories: care-providing tasks and administrative tasks. The first one involves the actual providing of care to a client. It involves washing and dressing clients, administering medication or performing treatment. The methodology for these tasks is highly variable and is nearly impossible to explore completely. It usually consists for a large part of manual labour, requiring personal skills. The second category consists of administrative tasks, like the registration of a client's personal details and insurance related data. The tasks belonging to this category are more structured and ideally should be performed without personal interpretation. This makes these tasks suited for automation. We will see more of administrative tasks in this domain later. In general, care-providing tasks are considered to be core-functions of any healthcare organisation where administrative tasks can be considered "supportive". Although administrative tasks are considered to be supportive, we have chosen to apply workflow technology only to the field of

[1] We will regard people with an age of 65 and above as those who are most likely to become dependent on healthcare

administrative tasks on basis that this field is more suited for automation.

Workflows can be seen as schematic representations of processes. What kind of processes is highly variable in nature, ranging from business workflows to scientific workflows. We will see more about workflow technology in chapter 3.

Topicus [4] is one of the larger providers of web-based services for the healthcare environment [2] in the Netherlands. Because the healthcare environment is complex and divers, Topicus was split up into several cells, which are smaller independent business units. Topicus Care, one of these cells, focuses on software applications for institutions for residential care and for organisations that provide care at the patient's private home. The domain of residential care will be our focus for this research.

Since home care and residential care (homes) are two similar worlds, we have chosen to eliminate the home care environment for this research. The first reason to do this is that there is a vast amount of procedures between these domains which are almost identical and could use the same solution. The second reason, however, is that I was requested to perform an in-depth exploration of supporting a process called the *Multidisciplinary Meeting*. This process exists on a regular basis within the residential home care environment, while it is nearly non-existent within home care.

This thesis consists of the following elements. First we will look at our healthcare domain in more detail (chapter 2). We will start by looking at the care-providing tasks and then move to the administrative tasks, as within the field of administrative tasks more profit from automation can be obtained. To obtain a better understanding we have chosen to describe the administrative tasks we had access to in the most detail. (Although the actual procedures might differ slightly, our example still provides a solid basis for further research.) After this overview of administrative tasks we will discuss some of the issues we came across within this domain. Based on these issues we will draw a conclusion on where we should focus our attention for support. The next section (section 2.3) describes the *Multidisciplinary Meeting*. Multidisciplinary meetings are held within our domain on regular basis, but are not regular business processes as they can be used either as evaluation or course determining process. Because of this we need to investigate the multidisciplinary meeting in more detail.

As mentioned before, we have chosen to use workflows as a basis for our automation. Therefore it is necessary to introduce workflow technology in more detail (chapter 3). In this chapter we will cover the general principles of workflow systems and look at the system we chose to use during our research (YAWL). In this chapter we will also investigate, based on the knowledge we gathered from the healthcare domain exploration, how we should apply workflow technology within this domain. After determining the workflow potential it is time to describe the prototypes.

In the chapter concerning the prototypes (chapter 4), we will first introduce concepts that were used during development. Then general layouts of the prototypes are provided where we will explain why we made certain choices during development. For our research we have chosen to create two different prototypes, however due to the time constraint only one of these prototypes could be developed into a working system.

[2] Topicus also provides web-based services for other markets, e.g. education and financial sectors.

We evaluated the first prototype that was created during this research (chapter 5). Three different evaluation sessions were organized to determine if our suggested solution with regards to the application of workflows would be received positively. During these sessions we focused on functionality, from an analyst's point of view, technical difficulties, from a developer's point of view and the overall potential. The last session was held with a combination of the two previously mentioned groups, as well as designers.

Next we will draw our conclusions (chapter 6). This is where all of the research questions are repeated and answered if they were not answered prior to this point. We will draw a conclusion about the usage of workflow technology within our domain and more importantly, we will determine if our suggestion of application is viable for larger use.

Finally we will look into future research and provide some comments on where the attention should be focused on (chapter 7). Here we found various interesting points during our research, ranging from practical issues we encountered during the development of our prototype and more theoretical issues with regards to workflow usage in general.

For the remainder of this chapter we will look at the research questions that were formulated as well as the methodology on how to obtain the answers.

1.1 Goals and Methodology

The focus of this research lies within the application of workflow technology in the healthcare domain (residential care homes). This section will provide an overview of the research questions involved and the methodology behind obtaining the answers.

1.1.1 Research Questions

For our research, we have chosen to create one main question and split it into several sub questions.

The main research question for this thesis:

- *How can workflow technology support processes found within the healthcare domain?*

The main research question as provided above is too large and should be narrowed down to fit in our timeframe. To limit ourselves, we have chosen to stick to the field of residential care homes. Additionally, we were asked to investigate one of the processes in more detail. This process is called the *Multidisciplinary Meeting* and will be discussed in the next chapter. To answer our research question we have chosen to create some sub questions. These are:

- RQ1: *What are workflows and what are their strengths and weaknesses?*
- RQ2: *What kind of processes can be found within the healthcare domain that could be supported by workflow systems?*

- RQ3: *What is a multidisciplinary meeting and what kind of processes does it contain that could be supported by workflow systems?*
- RQ4: *How can a workflow system be used to support the processes found by the second and third research questions?*

The remainder of this section will focus on the methodology we used to answer our research questions. It should be noted that our main research question should be supplemented with additional research into various other fields. For instance, we have limited ourselves to the residential care home domain, but within the domain of hospitals, the current situation might be completely different. In this research we have attempted to keep the basis as wide as possible and demonstrate where benefits can be obtained.

1.1.2 Methodology

As previously indicated, a comprehensive answer to our main research question cannot be obtained in our designated time span. To limit ourselves and put our focus on the healthcare domain Topicus Care works in, we have restricted our healthcare domain to only residential care homes. On occasion we will look beyond this domain for ideas and to see how certain issues are resolved. However we will always return to our limited domain.

The second research question (RQ2) will be answered by an exploration of the healthcare domain as provided by Van den Hoogen [27]. However, as this particular document was written in 2007 certain aspects might have changed. After having verified the correctness by Topicus healthcare experts it was found that the provided processes are still accurate enough to be used as they will only serve as an example to create the correct mindset. We have also arranged for an interview with André Middelkoop, section-director of the daycare department of the *Atlant Zorggroep* located in Apeldoorn, the Netherlands. This interview will be used to fill in any open gaps, but will primarily focus on the exploration of the multidisciplinary meetings (RQ3).

Since the processes found in our domain may differ between healthcare organisations, we have decided to make certain processes as abstract as possible. This means that we are not interested in the actual execution of a certain task, but rather to determine the "type" of processes which exist within the domain. With type we refer to the method of automation that we are considering, for instance, could a task (which is represented by a process) be completely automated or is human intervention required for the completion of a task?

After the introduction of the healthcare domain, we will focus ourselves on workflows. We will provide a basic overview of what workflows are and how they generally work. We cannot provide a complete overview of all the aspects of a workflow system, as many commercial and open source solutions exist and they all have different implementations. Most of them however, share a certain common believe in how a workflow system should be built and how the components should interact and that will be our partial answer to the first research question (RQ1). After we have clarified how workflows "work" we can determine what the strengths and weaknesses are of workflows. Once again this section is mainly theoretical because of the highly different implementations that exist today.

While investigating the applications of workflow we will also try to answer the fourth and final sub research question (RQ4). We will do this based on the strengths and weaknesses we have found as well as some of the functions workflows can embody. To test some of these strengths, we have developed a prototype that will demonstrate one of the functions that was found during our initial research. We have also included a prototype design (no actual implementation) of a second prototype which focuses on the ad-hoc creation of workflows as a response to the multidisciplinary meeting research. We will learn more about multidisciplinary meetings in the next chapter.

Chapter 2

The Healthcare Domain

As stated before, in the healthcare domain tasks can be divided into two different categories: care-providing tasks and administrative tasks. The first group is performed by healthcare specialists and involves direct interaction with a client e.g. the adjusting of a clients drip or medication. The second group is mainly performed by administrative workers, but is inevitable part of the job of a healthcare specialist. An example of an administrative task is requesting a new indication for a client. This chapter will serve as an example to illustrate what type of tasks could be found within our domain.

2.1 Care-providing Tasks

Care-providing tasks are highly divers by nature. Depending on the clients health these tasks can vary from administering medication to replacing bandages, etc. Because all tasks within this category are directly related to the health of a client certain issues with regards to workflow modelling can immediately be recognized. First of all, because all clients are different, each treatment is specific. Also healthcare specialists are never able to fully plan the recovery period of a client, especially when most of the clients in our domain are elderly. This makes making a perfect model to accommodate all clients very difficult. Suggestions were made to make a complete ontology for all procedures that can be found within the healthcare. This task was considered impossible because of the many variations. Certain specific healthcare specialties, like oncology, do use already predefined procedures, but are beyond the scope of this research. Because of the extra complexity in modelling care-providing tasks, we have chosen to first consider the administrative tasks because these are much more defined and static. Also, automation in the field of administrative tasks is only just beginning, especially with the use of workflow technology.

2.2 Administrative Tasks

This section introduces administrative procedures that can be found within the healthcare domain. We have chosen to use the research performed by Van Den Hoogen [27] as basis

for this section. Remaining gaps have been filled by an interview with André Middelkoop, section-director of the daycare department of the *Atlant Zorggroep* (Apeldoorn, The Netherlands). This interview was held to explore the concept of multidisciplinary meetings and will be discussed in more detail in section 2.3.

The following sections have been created to provide an extensive basis for discussing the type of processes in more detail. Although it may seem that the provided examples are too large and cover too much ground, we have chosen to include that particular section in this thesis to create a health-oriented mind-set. The following procedures do not describe an existing organization. We have chosen to describe these procedures in such a way that they are easy to understand, rather then trying to be as detailed as possible. An exact replication of healthcare business procedures would be impossible to accomplish on basis that each organization has developed a "slightly" different approach to performing these tasks. In short, we have tried to provide the best understanding of the processes that could be found, rather then try to be 100% accurate.

There are four different phases which can be distinguished (see also Figure 2.1) within the healthcare domain. In this section we will take a closer look at every phase. Please note that the given procedures in this section are based on information retrieved via various sources and therefore cannot be connected to one particular healthcare organisation. This section merely illustrates some of the possible procedures to provide a better context for the remainder of this thesis.

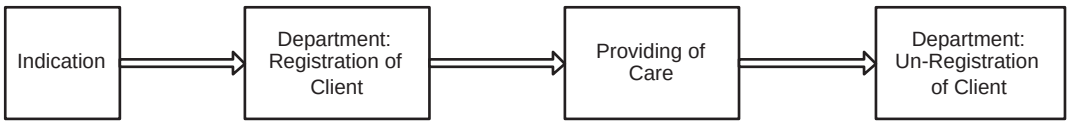

Figure 2.1: Overview of the four different phases, based on [27]

Today, transparency in procedures is highly valued within the healthcare branch. In the Netherlands awards can be obtained with regards to this transparency. As a result different health organisations have invited process experts to fully document all of their procedures. The *Atlant Zorggroep* has developed a near total overview of all its processes in the form of flow-charts. A Dutch example can be found in Figure 2.2 [1].

2.2.1 Indication

It all starts the moment a client makes a request for healthcare. This can either be done by the client himself or is done for him, for instance by his family doctor. A choice must be made between either moving a client into a residential home or organize care at the client's home. The steps that are taken are shown in Figure 2.3.

[1]Because of various reasons we were not allowed to include an actual copy of the procedure.

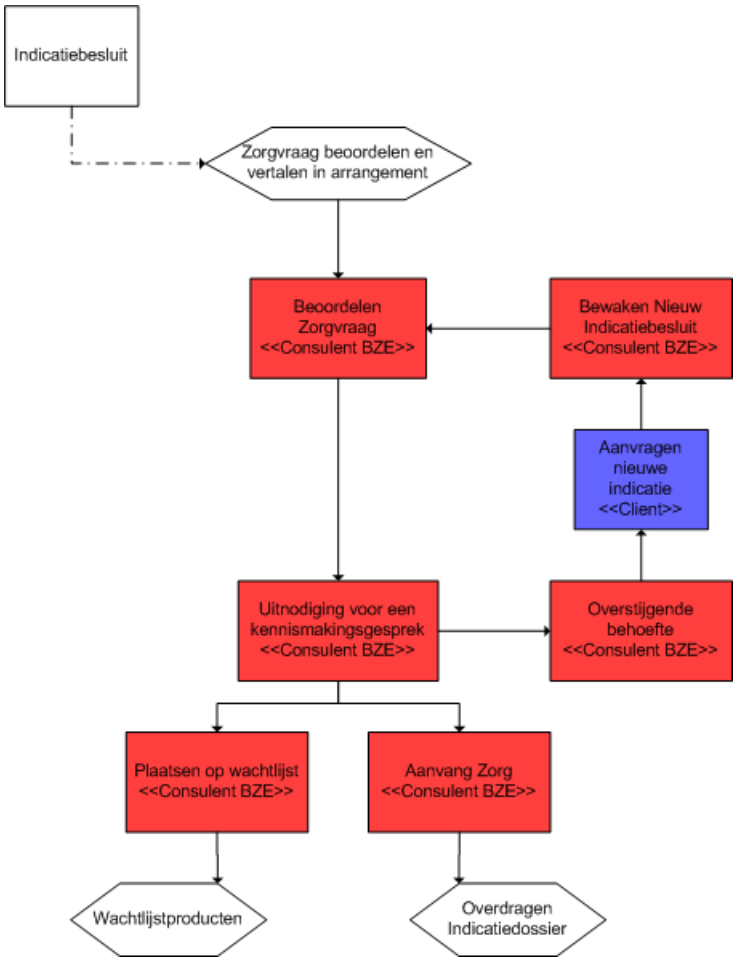

Figure 2.2: The procedure for introducing a new client after an indication was received, based on the official procedures found at the *Atlant Zorggroep*. The <<>> symbols describe (the role of) the person who executes the process.

Request Indication from *CIZ* An indication request is sent to the *Centrum Indicatiestelling Zorg* (CIZ). The CIZ is a governmental institute responsible for determining the healthcare requirements of the client and to determine how much money becomes available for this case. CIZ will judge the request and either accept or decline it. In case of acceptance the amount of care needed is indicated. This process could be supported by computer systems. However, this process is not part of the tasks of a care-providing organisation and therefore no part of this research.

In response to a positive decision the CIZ sends the indication to the healthcare providing organisation.

Add Client to Database The indication describes the level and amount of healthcare the client will receive. Usually each health organisation has certain pre-arranged "packages" (or products) of healthcare available. These products can be considered objects

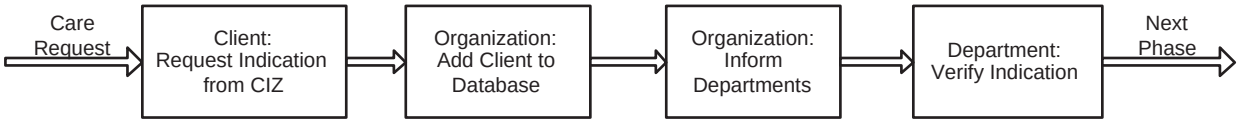

Figure 2.3: Indication phase, based on [27]

which can be bought and sold. Because there is no official standard for these products, each healthcare organisation has developed its own products. This means that there must be a translation from the indication to the products that will be bought. Once this is done, the client is added to the healthcare database. This is a process which can be supported by a computer system. However, because we are talking about healthcare implies that the final results of these translations should always be verified by an experienced employee.

Inform Departments Each healthcare organisation deals with an indication in a different manner. In some cases the previously mentioned translation is performed by another department than the one that will deliver the care. Other organisations might send the indication as a whole to the head of a department who will perform the translation. Regardless of procedure, the indication is moved down the organisational chain toward the healthcare department that will actually deliver the care. The department verifies it has the capacity to deal with a client and accepts (or declines) the new client. This process can be automated on basis that "informing a department" is nothing more than making a file accessible.

Verify Indication The translated indication is now accepted by a certain department. The indication request is now re-verified by healthcare specialists. This step is introduced to see if the previous indication is still up to date (sometimes a client's situation changes) or a new indication request must be issued. The verification must be done by a healthcare specialist, so it cannot be automated. The results however determine the next step: either a new indication is requested, which means this phase is started over, or we proceed to the next phase. An automated system to handle the results of these evaluations could be created.

2.2.2 Registration of Client

At this point, the initial administrative and financial tasks are completed. This phase prepares the client for the receiving of the actual healthcare. A schematic overview of the processes in this category can be found in Figure 2.4.

Introduction of Client The healthcare organisation introduces the new client to its staff. One of the methods used to do this is sending out a memo that a new client has been taken on by the organisation. This can be automated by a system that notifies certain members of staff (e.g. department heads) by means of an automatically generated message.

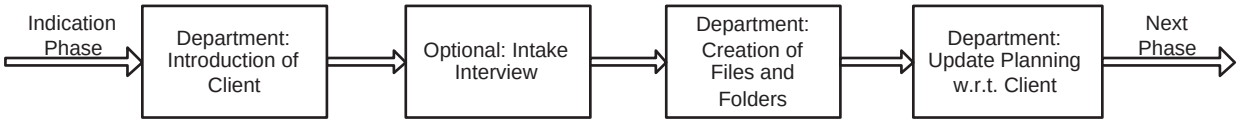

Figure 2.4: Department: Registration of Client, based on [27]

Optional: Intake Interview In some cases it is necessary to perform an intake interview, which focuses on any specific demands or requests a client has. The intake interview is then added to the client's file, which is created next.

Creation of Files and Folders The process of health providing is one that is continuous and constantly re-evaluated. This produces a vast amount of paperwork which must be stored properly. The *Atlant Zorggroep* uses paper folders which are filled with template documents of events that will occur. This is an administrative job that can be automated. There is however one problem with facilitating this step. By Dutch law, all healthcare organisations are obligated to keep a paper copy of each client's file. So, there are rather few healthcare organisations that are fully automated at the moment of writing. Since there must be a paper copy of all documents, why bother digitizing and printing them?

Update Planning w.r.t. Client Now that a client is registered as in need of healthcare, time must be reserved to actually provide the care. This is done by means of planning. The planning of healthcare employees w.r.t. their clients is a process which could be facilitated by means of a planning program. Because planning algorithms in general are quite difficult and our focus for this research is kept as wide as possible, we have chosen not to worry about planning situations within our domain. This reason also applies to other planning-related processes we will see later in this chapter.

At this moment the client is, from an administrative point of view, receiving healthcare. However, specialists are obliged to create a personal health plan. This plan is only valid for about 6 weeks after which a new client should be discussed in a multidisciplinary meeting. The multidisciplinary meeting will be discussed in section 2.3.

2.2.3 Providing of Healthcare

The providing of healthcare is a continual cycle of a wide range of activities. Van den Hoogen describes eight different tasks, which will now be discussed. Because all of these processes can happen at random moments, several times or not at all, we cannot provide a diagram of the processes in this category.

Request New Indication If, at any moment the indication given for the client is no longer sufficient (e.g. the client needs more care than prescribed) a request for a new indication is send. As soon as the new indication is received, the clients planning is updated to provide for the new healthcare situation.

Register Hours The care providers have to register how much time they have spent on a particular client (and in total). This is a part of the financial administration. At this moment the automatic registration of hours spend by a healthcare professional is one of the most popular to facilitate. Several systems already exists which support this process (and the related processes), with a wide range of tools at their disposal. Some of these systems use workflow technology, others do not. Therefore we have decided not to include this particular process in this research.

Answer Emergency Calls Most healthcare organisations provide a system or (at least) a phone number which can be dialled by clients in case of an emergency. Because the actual handling of an emergency is time critical, this task is not included in this research.

Work Preparation When a care expert is going to have contact with a client, he has to know about latest developments concerning this client. This is also true when the expert is going tot attend a multidisciplinary meeting. Every person has his own way of preparing for a meeting. Some of the interviewed staff stated that they always have a look at client's file before doing anything. Others state that they "already know" what the situation of a particular client is and do not need to prepare at all. A possible method of supporting work preparation is to provide staff with the option of getting all information with one click of a button. This means either printing the whole document or displaying a certain section as the staff member desires.

Update Files The updating of a client's files is a continual process. First of all it must be done to inform other healthcare specialists of what has happened to a client in their absence (e.g. what kind of treatment did a client receive). Additionally, healthcare organisations are obliged by law to register all actions related to the client. This registration is done to see who is responsible in case something goes wrong. Finally, by registering the amount of hours that is spent on a client, the client's insurance company can be charged for the delivered healthcare. Depending on the current situation of a healthcare organisation (are they using digital client files or paper versions) this is a step that can be facilitated.

Evaluate Client Situation The evaluation of the clients current situation is a continuous process. The goal of this step is to determine whether the clients healthcare plans should be adjusted or potentially a new indication should be requested. The outcomes of these evaluations are added to the client's file and discussed during (multidisciplinary) meetings.

Re-planning of Care Providing Although most care providing is done in regular intervals, in some cases it is necessary to adjust this planning. Reasons for this re-planning can vary, ranging from staff shortage to a new indication prescribing another healthcare package. Planning is a part of the healthcare system we have chosen not to include, although facilitation in this area is very well possible.

Multidisciplinary Meetings Multidisciplinary meetings are held for every client at least twice a year (as described by Dutch law). The goal of multidisciplinary meetings is to evaluate the current healthcare position of a client and act accordingly. This step is discussed in more detail in section 2.3.

Most of the processes mentioned before happen at irregular intervals. They can happen in parallel or sequential, depending on the status of a client. E.g. in case a client gets ill, the need for healthcare increases, emergencies are more likely to happen, etc.

2.2.4 Un-registration of Client

Eventually all clients will enter the un-registration phase. In this phase the providing of care in its current form is halted. In general, there are three different reasons to enter this stage: the client no longer requires healthcare, the client's health worsens and the need for care increases or the client died. Nevertheless, the current healthcare agreements become void and the client has to be un-registered. The following procedure is used to un-register a client. A schematic overview can be found in Figure 2.5.

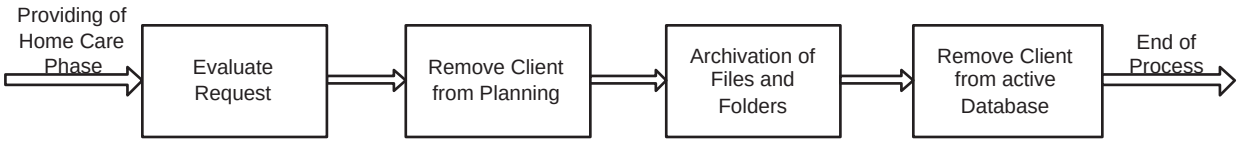

Figure 2.5: Un-registration phase, based on [27]

Evaluate Request Because the provision of healthcare should never be stopped all of a sudden, the request is evaluated by a specialist. There is no real facilitation possible for this step, but it should be noted that some processes within the healthcare system needs additional security.

Remove Client from planning Once it is confirmed that the client no longer requires healthcare, the client's data is removed from the planning to make the hours available for other clients. This action is directly connected to the planning process and will not be discussed in this research.

Archiving of Files and Folders The client related files and folders are archived for later use. Also by Dutch law healthcare organisations are obliged to keep all their files and folders in archive for several years. As mentioned earlier, the level of facilitation for this action highly depends on the current level of IT present in the healthcare organisation. In case client files are fully digitized (which presently non are, due to Dutch law) then this step could be fully automated.

Remove Client from active Database The healthcare organisations updates its database. This client is no longer receiving healthcare.

2.2.5 Current Issues

In her thesis Van den Hoogen also discovered some unresolved issues. These issues originate from 2007 and may already be resolved in a particular instance. In general, these problems still exist today. This section will look at some of the mentioned issues.

Information Gets Lost The (paper) file of a client usually resides in one place. But a certain specialist may take the client's file to some other place, rendering it inaccessible to other employees. Strict rules and regulations are usually in effect to prevent the loss of a client's file. Still it happens.

Information Not Available In addition to the first problem, sometimes certain pieces of information might not be present at the correct place at the correct time. For instance, it could be that client's file is taken to be updated while someone else needs to check certain data before performing medical treatment.

No Overview Of Total Delivered Care Keeping track of what the client already received in forms of healthcare is something that needs to be done to receive funding. All provided healthcare is (usually) funded by the client's insurance company and it requires that a record is kept. Based on that record the healthcare organisation receives funding to proceed with the healthcare provision for a certain client.

Double Storage To resolve the previous mentioned issue of information not being available, some healthcare organisations have opted to create duplicate client files to make the information more accessible. The introduction of duplicate files means that the synchronization of these files becomes an issue which in some situations can lead to serious errors.

Duplicate Process Execution In addition to the previous described gap of information, it might be the case that certain healthcare processes are repeated because the registration of such action was not complete.

First, it should be noted that the previously described processes can be found in an environment where no IT system is present. (The goal of Van den Hoogen‘s thesis is to determine how and where IT could be introduced into the current situation.) Most of these issues therefore can be resolved by the introduction of an IT system which keeps the storage of client dossiers centralized. In this research, we are only interested in the application of workflow technology in situations where an IT based system already exists. The problem involving the lack of overview of total delivered care can be solved and we will see more about that in the next chapters.

2.3 Multidisciplinary Meetings

This chapter focuses on multidisciplinary meetings (MDM's). An introduction to MDM's is given, followed by an exploration of the group formation and procedures concerning MDM's.

2.3.1 Introduction to Multidisciplinary Meetings

Multidisciplinary meetings exist in various settings. They exist in large corporations, where the multidisciplinary part refers to different departments sending over their people to attend a meeting. Healthcare-related MDM's mainly exist within hospitals. Hospital-based MDM's consist of specialists who share a certain interest or goal. Hospital-based MDM's are usually organized to solve a problem, ranging from creating new procedures to determining a plan of treatment for a certain patient.

This thesis focuses on the residential care environment and here a totally different kind of MDM exists. While in hospitals the goal of MDM's are to resolve a certain problem (e.g. how to treat the patient), a MDM in our domain is held to discuss the progression of the client. During these MDM's, the current healthcare situation of a client is evaluated, looking at current condition and possible changes a client will experience. In this setting, there is no real "goal" which can be attained. Prior to each meeting the primary caretaker evaluates the current situation with the client or a representative of this client in case the client is unable to evaluate for himself.

In accordance to Dutch law, each client in a residential care home is discussed at least twice a year. Additional meetings can be organized whenever it becomes necessary.

For the sake of simplicity we will use the term "client" to refer to a person who can evaluate the current healthcare status of the receiver of healthcare (former definition of client). This means that during this chapter the "client" can be the actual client or a legal representative (partner, child, etc.) of the client. In case neither the actual client nor a relative can attend, the primary caretaker (see below) will take over this responsibility.

2.3.2 Group Formation

Although the groups formation for a MDM depends on the clients healthcare packages, a "core" group of people can be distinguished who, in theory, should always be present at a MDM. This section briefly introduces the different persons, along with their roles within the meeting.

Client In contrary to hospital-like MDM's the client plays a key role in his own treatment plan. The role of the client is to be the informer. In case the client is unable to attend, the primary caretaker informs the person in his care as soon as possible after the meeting.

Primary Caretaker Each client is assigned a primary caretaker at the moment he is introduced in the healthcare organisation [32]. The primary caretaker is responsible for organizing the meetings. He is also responsible for all administrative actions related to these meetings. Although he does not have to perform all necessary activities himself, he is, in the end, responsible for the correct completion of all these tasks. In case the client is unable to make decisions or unable to attend a MDM, the caretaker will take over his role as informer and forward the outcome to the client as soon as possible.

Primary Physician The clients primary physician is responsible for the medical treatment of the client. He regulates the medicine use of a client and determines other treatment plans when needed. The primary physician is not always present at a client's MDM due to a busy schedule but tries to be present as much as possible.

Relevant Specialists In case the client is treated by different specialists (other than the primary physician) the presence of these specialists is desired. In case it is not possible to attend a meeting in person, the different specialists are asked to provide written input to the meeting.

Meeting Leader / Chairman Each MDM has a chairman. This cannot be the primary caretaker, primary physician or the client for obvious reasons. The *Atlant Zorggroep* appoints the department head to be the chairman. The chairman's responsibility is to guide the discussion, keep the meeting on track and make sure that all necessary points are discussed within the given time constraints.

2.3.3 Procedures

As there are no official guidelines or rules for a MDM, healthcare organisations have developed their own method of organizing these meetings. The *Atlant Zorggroep* has developed an in-house overview of the procedures regarding MDM's. We will take a look at these procedures and try to determine where we can offer support to the prescribed procedures.

Pre-meeting procedures

First the primary caretaker sends out invitations for the meeting suggesting a suitable time and date. These invitations are sent to all involved.

At least 3 days before the actual meeting each healthcare provider sends in a brief summary of all healthcare related actions he has taken with regards to the client. Along with this summary each provider has the possibility to address certain issues he deems relevant to discuss during the meeting.

The client also has the possibility to send in issues to be discussed. In case the client was discussed by a MDM prior to this one, the primary caretaker will review all issues from the last MDM with the client. This is done to determine if all issues from last MDM are resolved or that additional actions must be taken. The result to this contact between the client and the primary caretaker can be seen as *the* input for the upcoming MDM.

All attendees are expected to have knowledge of the latest updates of the client's case file before the meeting.

Meeting Procedures

According to the interview there is little ICT support necessary during the actual meeting itself. During the meeting the focus lies on the social interaction between the client and his

caretakers. Because of this we have chosen not to "enforce" ICT solutions into this meeting, but rather focus our attention on providing support outside the meeting.

Post-meeting procedures

It is the responsibility of the primary caretaker to update all the clients files with the results of the meeting. Furthermore it is common practice to include a separate report on the MDM itself in the client's file. In this document all discussed issues and decisions are recorded. This list will be taken by the primary caretaker to the client for evaluation, which serves as the input for a future MDM.

All other specialists are responsible for their own treatment plans and the updating of the client's file accordingly. Specialists who were not present at the MDM are supposed to update themselves with the clients status by reading the client's file.

2.4 Summary and Conclusion

In this chapter we have examined procedures found in both the general administrative domain as well as in multidisciplinary meetings. To provide the best ideas with regard to the processes found, one extensive example on what type of processes could be found is included. During this examination we have excluded certain process types, like planning problems and emergency responses. These types of processes were left out because of the complexity (or importance) that these processes bring with them. These processes require additional research and our original intentions were to treat as many different processes as possible to provide a wide basis on which we can base our support suggestions.

The multidisciplinary meeting has been discussed where we have focused on the group formation and on the procedures prior and after the meeting itself. Before and during the meeting we have found no suitable candidates to be supported (some if we are getting very creative) but within the post-meeting procedures we have found some possibilities. The processes best suited to be supported are those with regards to incidental task scheduling, e.g. the assignment of non-repetitive tasks to certain healthcare professionals. We will see more about supporting this particular kind of ad-hoc tasks later.

With regards to the general administrative tasks, we are not surprised that the dominant process type to be supported is that of traditional data processing. Many processes involve either the manipulation of existing data or the insertion of new data. In the next chapter, when looking at workflow systems, we will return to the question on where support can be provided.

Chapter 3

Workflows

In order to answer our research questions we have to take a good look at the workflow itself. In this chapter we will explore the domain of workflow systems and learn what we need to know to be able to create an application for workflows within the healthcare domain. We will first introduce the concept of workflows, see what they are generally used for and how they behave. Because workflow systems vary from system to system we will then introduce the "general" form workflows usually come by. We will look at the mechanics of a workflow starting with a global overview, then move on to the workflow model and finally say something about the actual running of a workflow.

The second part of this chapter consists of the exploration of workflow uses. We will determine functions which workflow systems can fulfil followed by a determination of strengths and weaknesses of workflow systems. We will then use this information and determine the workflow potential in both general use and in the multidisciplinary setting.

We will end this chapter with an overview of some of the related workflow systems we have found for our domain. First of all, we will introduce the YAWL workflow system which was used for this research. Although the introduction to workflows will be more general, some parts are rather YAWL specific. Therefore we have chosen to introduce YAWL and the reasons why we have chosen for this particular workflow system, first.

3.1 Introduction to Workflows

A fundamental question that came up during this research was about the usage of workflows itself: Why should systems use workflow technology and what is there to be gained by using them? Without going into too much detail - we will come back to that later - a workflow system should be seen as a structuring of tasks according to a certain model. By using a model at the basis of the system a workflow system provides a certain level of freedom unprecedented by other systems. Leymann and Roller [15] state that the usage of workflow systems changes the perspective of applications from a data (or flow) driven approach to something that uses models and activity implementations. We reckon that this approach suits our needs best for the task-driven environment we want to create within our healthcare

domain.

However, before exploring this task-driven environment, we should explain how workflows work in more detail. Before starting, mind that there are two different types of workflow: scientific and business workflows. These different workflow systems are discussed next.

3.1.1 Scientific Workflows

Scientific workflows are *"networks of analytical steps that may involve, e.g., database access and querying steps, data analysis and mining steps, and many other steps including computationally intensive jobs on high performance cluster computers."* [17, p 1] Their focus lies on the analysis of data streams [18]. For this thesis, scientific workflows will be considered to be more execution-focused than guiding. A scientific workflow describes in detail how processes should be executed. The idea is that (nearly) all work is accomplished by a computer. Good examples of scientific workflow systems are MyGrid's Taverna [26] and Kepler [14].

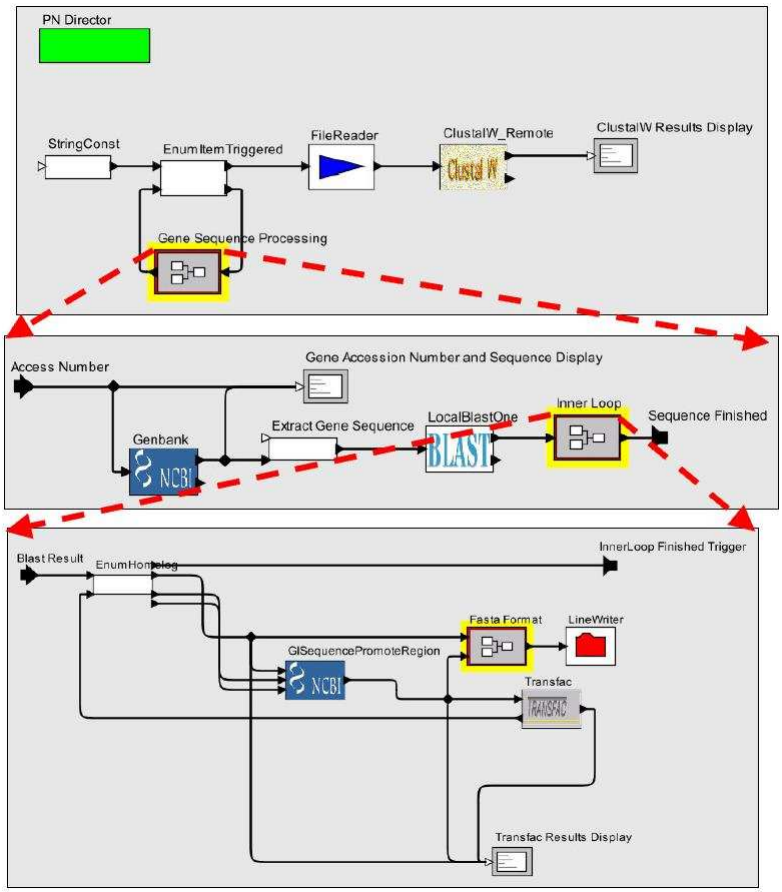

Figure 3.1: A scientific workflow, obtained from [17]. It demonstrates how a process from a workflow model can be unfolded to a completely new workflow model. This is useful to create "complex" processes.

In Figure 3.1 we see a scientific workflow which is built from other scientific workflows (workflow processes can be other workflows). In general, scientific workflows are mainly data-driven, meaning that they are built of pieces of code performing certain actions on the data streams they receive. For our research we consider scientific workflows not the best match for our goals. Therefore we will not investigate the application of scientific workflows in more detail.

3.1.2 Business Workflows

According to the Workflow Management Coalition (WfMC) [11], business workflows are *"the automation of a business process, in whole or parts, where documents, information or tasks are passed from one participant to another to be processed, according to a set of procedural rules."* This means that a business workflow is used to control information paths and processes. It is able to generate tasks which are sent to users and it handles the results coming from those users.

For this thesis, we will only consider business workflows because within residential home care a substantial part of all tasks must be performed by people. The next section will cover the basic mechanics of a (business) workflow system.

3.2 Introduction to YAWL

The complete YAWL[1] system [8] originated from the Eindhoven University of Technology and Queensland University of Technology and can be considered to be a business workflow system [2]. The YAWL system was designed after a careful review of over 30 different other workflow systems and languages and is based on the works of the workflow pattern initiative [29]. Including in the complete YAWL system is the workflow engine and the YAWL language, which is used to store workflow models. Van der Aalst et al. [29] describe the YAWL system on basis of discussions regarding the implementation of different perspectives. It provides a good overview of the complete workflow system, including all concepts related to the use of workflows in general. In this section we will introduce YAWL on a deeper level by looking closer into the workflow patterns on which it is based. Also, we will look at Petri Nets that form the foundation to use these patterns.

Workflow Patterns Workflow patterns, functionality described in the form of a model without regards of the actual implementing language, were originally designed to provide a basis of comparison between different workflow systems. As Van der Aalst et al. [30] quote about their paper: "*This paper can be seen as the academic response to evaluations made by prestigious consulting companies*"[30, p 1]. Van der Aalst et al. have created several workflow patterns for modelling constructs that occur often. From a higher perspective these

[1] YAWL stands for *Yet Another Workflow Language*

[2] Although a certain level of programming freedom is provided to the user, it is designed to be a business workflow as stated in [29]

constructs can be classified into different categories or "perspectives" as they are called. The *control-flow perspective* is used to describe the ordering in which tasks should be accomplished (e.g. sequence, choice and parallelism). The *data perspective* and *resource perspective* are used to describe the flow of objects. In the data perspective we see document flows and sometimes workflow environmental variables. The resource perspective is used to describe humans and/or device roles responsibility with regards to a certain task. Finally we have the *operational perspective*, which describes the elementary actions executed by the tasks that a model describes. (Usually data is transmitted between the workflow system and some external data-manipulating application. In such situations it is useful to have information about what actions are being performed outside the workflow system as well as on the inside.) The remainder of [30] describes a series of patterns which are used for the comparison of workflow systems.

Petri Nets At the basis of the YAWL workflow systems a concept called *Petri Nets* can be found. Petri Nets are one of the possible modelling languages used for describing discrete distributed systems [3]. Petri Nets are used for several reasons. These reasons are given below and are directly cited from [28, p 3-4].

Formal Semantics *A workflow process specified in terms of a Petri net has a clear and precise definition, because the semantics of the classical Petri net and several enhancements (colour, time, hierarchy) have been defined formally.*

Graphical Nature *Petri nets are a graphical language. As a result, Petri nets are intuitive and easy to learn. The graphical nature also supports the communication with end-users.*

Expressiveness *Petri nets support all the primitives needed to model a workflow process. All the routing constructs present in todays workflow management systems can be modelled. Moreover, the fact that states are represented explicitly, allows for the modelling of milestones and implicit choices.*

Properties *In the last three decades many people have investigated the basic properties of Petri nets. The firm mathematical foundation allows for the reasoning about these properties. As a result, there is a lot of common knowledge, in the form of books and articles, about this modelling technique.*

Analysis *Petri nets are marked by the availability of many analysis techniques. Clearly, this is a great asset in favour of the use of Petri nets for workflow modelling. These techniques can be used to prove properties (safety properties, invariance properties, deadlock, etc.) and to calculate performance measures (response times, waiting times, occupation rates, etc.). In this way it is possible to evaluate alternative workflows using standard Petri-net-based analysis tools.*

[3] A formal description of Petri Nets and how they function can be found in [19]

Vendor Independent *Petri nets provide a tool-independent framework for modelling and analyzing processes. Petri nets are not based on a software package of a specific vendor and do not cease to exist if a new version is released or when one vendor takes over another vendor*

In this research we have not used any "special" features for which YAWL is well-known, e.g. the use of workflow patterns. The YAWL system was chosen for different reasons we will discuss next.

3.2.1 The Choice for YAWL

This section describes the reasons why we have chosen for the YAWL system in favour of other, sometimes more matured, systems.

First of all, the YAWL workflow system is a workflow system designed to handle multiple users. In the domain of residential care, the first thing mentioned is that all information in a client's dossier is sensitive and private and should only be read by people who have to know about these facts. This indicates that separation of different users is desirable. (Privileges and rights.) Most workflow systems are unaware of different users. In fact the YAWL workflow engine is as well, but the YAWL system is distributed by default with a resource manager. This resource manager provides a basic user interface and allows for the creation of different workflow users and roles.

Secondly, the YAWL workflow system is an ongoing research project which has called out to be used in real life situations. As this thesis is about a real life situation we have chosen to prefer YAWL over commercial workflow engines. YAWL is an open-source, well supported and free of charge workflow system and is mentioned in a large amount of publications about workflows and their applications in general.

Thirdly, YAWL was already known at the Topicus company. The YAWL workflow system was already attempted to be integrated on several different projects within Topicus with variable success. Although it was never used in any commercial products, certain employees have used the YAWL workflow system for experimentation to resolve certain problems. Because of this experimentation we were asked to, if possible, use the YAWL workflow system.

The fourth reason is that YAWL was written in the Java programming language. The Java programming language is taught extensively at the University of Twente and is one of the best known programming languages to us.

3.3 Workflow Mechanics

In this chapter we will cover the inner workings of workflow systems in more detail. The term "workflow" comes with a large amount of definitions and terminology. As Song et al. wrote in [23, p 1]: "*the semantics behind the terminologies as they are intended by the different parties are rarely the same (e.g., what do you actually mean by "workflows".)*" During this research we have experienced this problem in terminology, therefore it is vital to clarify some of the definitions. We will do this in the following sections.

In this chapter we will look at the different semantics used in this thesis. To create our definitions, we have used various sources. In some cases however, we were forced to adjust some of the terminology (like we did for healthcare in the previous chapter).

As stated before, one tends to think of workflow systems as "just" the workflow model. We will look at workflow modelling first. Next we will place the modelling into a larger context and look at the complete workflow system. Then the workflow system we have chosen to use for this research will be introduced. Finally we will look at how workflow enactment, the execution of workflow models, is done within this system.

3.3.1 Workflow Models and Instances

When talking about workflows people tend to think that workflows are just drawn graphs. However, the official definition, as provided by the Workflow Management Coalition, is that a workflow is *a computerized facilitation or automation of a business process, in whole or part.* [11, p 6]. When people use the word "workflow", they are usually referring to the "workflow model" or "process definition". The workflow model is the representation of one (or multiple) process definition(s). According to the Workflow Management Coalition, a workflow model (or process definition) is *the computerized representation of a process that includes the manual definition and workflow definition* [11, p 7]. An example of a simple workflow model can be seen in Figure 3.2.

Workflow models are generally created by means of a workflow editor. Numerous editors are available, depending on the data format in which a workflow model is expressed. The YAWL system uses a flat XML data structure and can be edited by means of the workflow editor tool, which is provided with the distribution.

Workflow Processes Processes are the "working elements" of a workflow model. They can represent numerous items, some more formal than other. Remember the difference between business workflows, where processes can describe certain organisational units versus scientific workflows, where processes represent certain data filters or operations. What a process actually entails is highly dependent on the type of workflow system that is being developed. One question that should be answered is that of which role the workflow system should fulfil within the complete system. The role that a workflow system can be given is directly related to the type of processes that are present within the environment. Becker et al. [2] state that processes can be divided into three categories, namely *organisational*, *software* and *hybrid* processes. Organisational processes are mainly human-driven and workflow models are mostly used to describe the routing of documents. Software processes use workflow systems as "glue" to connect different separate systems and how certain outputs must be treated as inputs for other systems. Hybrid processes combine the former two and use workflow systems as *"an intermediary between the human participants and a (functionally oriented) application system [...] guiding the work of participants within single activities"* [2, p 42].

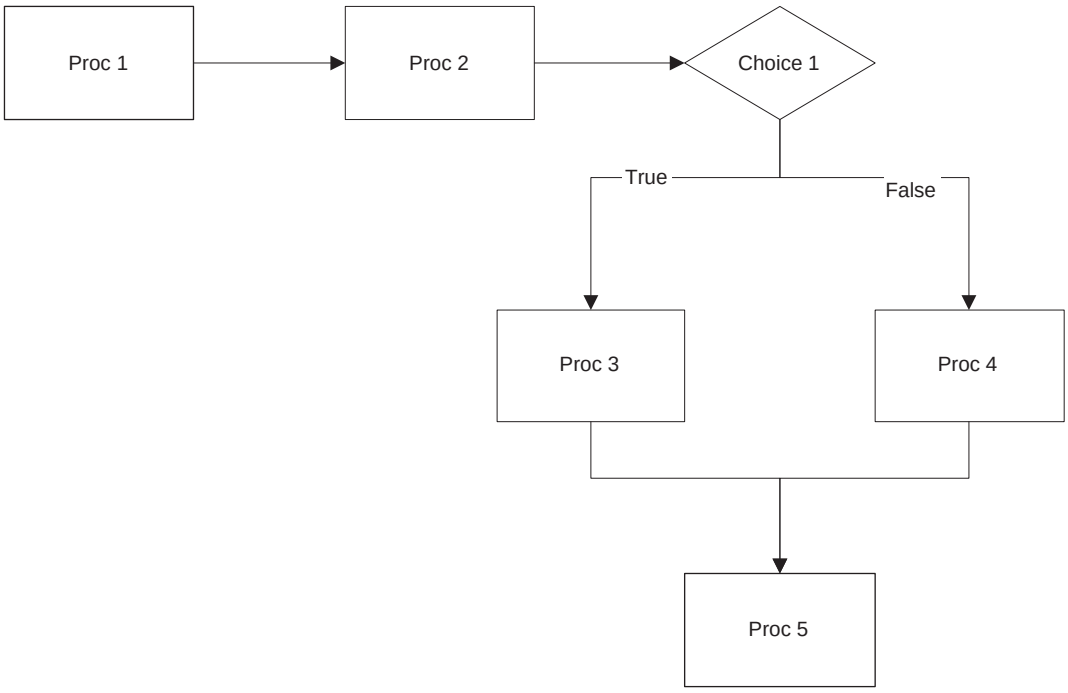

Figure 3.2: A very basic workflow with one choice. The workflow model should be read as: *The process begins at "proc 1", which is followed by "proc 2". At this moment we need to make a choice "Choice 1" and depending on the outcome we execute either "Proc 3" or "Proc 4". Finally we execute "Proc 5".*

Workflow Processes as Graphs From a more theoretical point of view, we must acknowledge that workflows are graphs. Graphs are mathematical structures used to model relations between objects "from" a certain collection. A graph consists of vertices (the objects, usually represented as circles, squares, etc.) and edges (the lines between the vertices). Graphs can either be directed or undirected, meaning that the edges either point from one object to another (directed) or do not provide a direction at all (undirected). Within mathematics, there a lot of algorithms which use graphs as basis, including transformation (the manipulation of the structure of a graph) and traversal (the exploration of a graph) which could be used for the analysis of a workflow model on a mathematical level. For more information about graphs, how they function and example algorithms, we would like to refer to [25, 22].

Workflow models are directed graphs. These directions describe the flow of a workflow, e.g. in which order the processes need to be executed. Because workflows behave as directed graphs, we can apply algorithms to graphs whenever we want to. This means that we can perform certain analysis on the structure of the graph, rather than the actual contents of a workflow. From this point on, we will assume that all operations on the structure of a workflow model can be accomplished by manipulating it with graph algorithms. In section 4.2, we will show why this is very useful.

Workflow Instances Without understanding the actual workings of a workflow system, one can imagine that a workflow model eventually is "run". Formally this is known as workflow *instancing*. A workflow instance is a piece of code that is being executed by the workflow engine, it can be compared to object instancing we know from objected oriented programming: a memory copy of the workflow model is created and then "run".) A workflow instance can be in different states of execution. These different states are displayed in Figure 3.3.

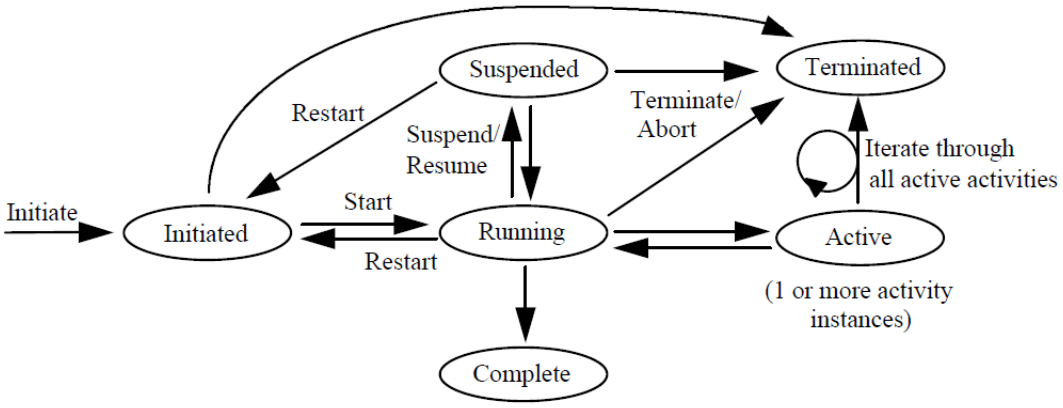

Figure 3.3: The different states of a workflow instance, obtained from [11]. At the moment an instance is initiated, the instance enters the "initiated" state.

A workflow instance is a working copy of the workflow model. In this thesis we will not go into the actual mechanics behind instancing a workflow model.

3.3.2 The Complete Workflow System

As the previous section mentioned, *workflows* are generally misunderstood and when talking about workflows people generally mean the workflow model. However, the workflow model is only a part of the total workflow system. In this section we will take a closer look at the components which form a workflow system. In a previous section we have also described the difference between a scientific workflow and a business workflow. From these differences we must conclude that there is not just one solution to "what a workflow is". The description given in this section is largely based on the works of the Workflow Management Coalition (WfMC) [11] and Van der Aalst et al. [29]. The first describes business workflows in detail and van Aalst et al. have developed the YAWL Workflow system which has been used during this research. We will return to YAWL in more detail later in this section.

We will first systematically describe a workflow system in detail. Next we will look at the communication between the workflow system and "others" and finally we will look at how a workflow is actually executed.

Workflow Systems

The overall workflow system, including all components and separate modules, is called a workflow management system (WFM). A WFM system is *a system that completely defines, manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic* [11, p 6]. An overview of a workflow system can be found in Figure 3.4. We will return to these interfaces shortly.

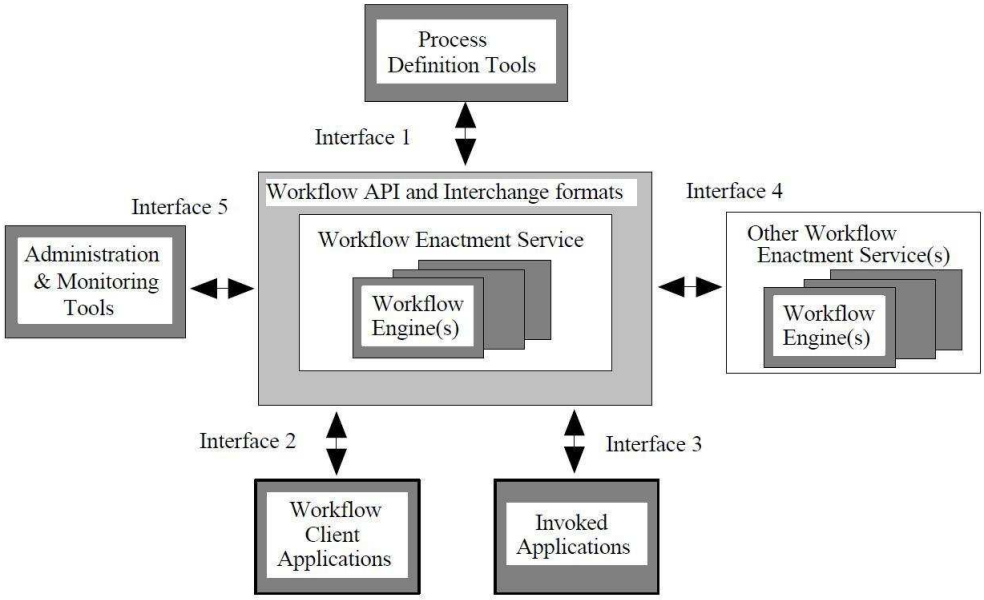

Figure 3.4: A Workflow System, obtained from [11]

Workflow Enactment Service At the core of a workflow system lays the workflow enactment service. This service consists of one or more workflow engines and is solely responsible for the creation, managing and execution of workflow instances. The workflow enactment service also provides interfaces to the underlying workflow engines on basis of the workflow application programming interface (WAPI). An application programming interface (API) is an interface created by the developers of a certain program (in our case a workflow system) to allow other software to interact with the existing system. The "enactment" of a workflow model is simply the uploading of the model to a workflow engine and instantiating it.

Workflow Interfaces A WFM system is composed of various modules. Some are called interfaces. Interfaces can be seen as access points to workflow systems, each with a specific purpose. The interfaces shown in Figure 3.4 provide display interfaces which are usually offered by workflow systems to system developers.

These interfaces take care of the following:

- Interface 1 - This interface allows the integration of a process definition tool.

- Interface 2 - A workflow client application allows the manipulation and advanced control of worklists.
- Interface 3 - (Remote) invoked applications can be used to fulfil tasks the workflow was not designed to accomplish. (E.g. graph generation or direct database accessing)
- Interface 4 - Other workflow enactment services can be addressed in case certain processes use remotely executed workflows themselves.
- Interface 5 - Administration and monitoring tools allow the system to be controlled and monitored.

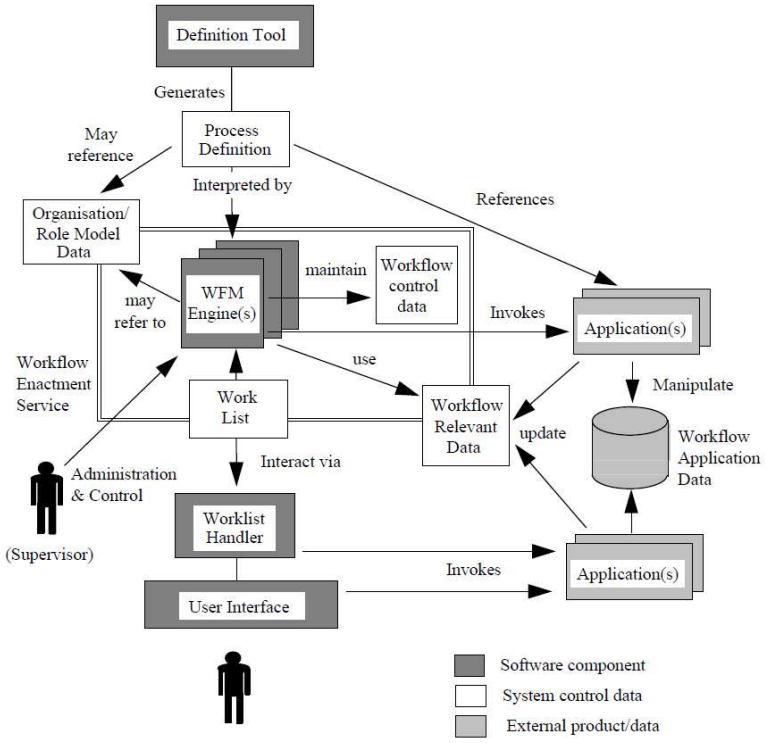

Figure 3.5: A workflow management system in its environment, obtained from [11]

Figure 3.5 shows an implemented workflow management system, with a more detailed view of what a workflow enactment service actually entails. For instance, interface 2 is implemented by a worklist handler. We will now look at the definitions of this specific part of the workflow management system.

3.3.3 WorkItems, WorkItemRecords and WorkLists

This section describes different concepts used by the YAWL workflow engine. Although some of these concepts are not unknown to other workflow systems, there is no guarantee

that the actual implementation of the concepts described in this chapter are all the same. Since we have chosen to use the YAWL workflow system for our research, we have included this section into our report. The terminology used in this section originates from [3] unless specified otherwise.

WorkItems At the moment a workflow is executed, processes are launched and a queue is formed of instructions that need to be performed. These instructions are known in the YAWL context as *WorkItems*. A WorkItem can be seen as link between a case (or workflow instance) and task [28]. The workflow system itself is unaware of and has no interest in the method of execution of the workitems. This means that a workitem can represent some human action, some pre-programmed script or a call to a completely different system. To a workflow system, all these different representations are similar.

WorkItemRecords WorkItems contain some code that can be considered the "core" of a workitem. This core is known was the WorkItemRecord. The workitemrecord is a record that contains crucial information for the workflow engine. An example of a workitemrecord can be found below. Note that some of the information is omitted for demonstrative purposes.

```
<workItemRecord>
  <id>975:Start_Verwerking_4</id>
  <caseid>975</caseid>
  <taskid>Start_Verwerking_4</taskid>
  <uniqueid>00000000000000000000000003</uniqueid>
  ....
</workItemRecord>
```

We are not concerned with the actual inner workings of a workflow engine. However, we need to point out a few of the attributes used within workitemrecords that will play an important role during the development of our prototypes. First of all, the *caseid* is a numerical representation of the workflow instance that this workitemrecord belongs to. This attribute is used to make a distinction between different clients in our system. The second attribute that should be noted is *taskid*. The *taskid* represents the process described within the workflow model. We will see more of the workitemrecords in the description of the prototype. These two attributes combined form the overall *id* which the workflow systems uses as a global identifier. Although this identifier connects the case to a certain task it does not necessarily make it unique, since one process can exist multiple times within a workflow model. Because of this, we cannot reference to a certain workitemrecord with complete certainty and this could become a problem. Therefore the *uniqueid* attribute was created. We will not discuss the reasons why someone would like to make individual workitemrecords accessible, but just state that from a programming perspective this would be highly useful.

For now we will leave the workitemrecords and focus on their application and usage.

WorkLists WorkLists are containers in which workitems reside. The YAWL workflow system contains several different worklists each representing a certain state of the workitem. Worklists come with a rather limited set of functions to retrieve and search workitems and are the basis on which a worklist handler is created. At the creation of our first prototype we will see some of these functions in action. For now it is enough to know that worklists are the containers of workitems on which certain operations can be performed.

3.4 Workflow Application

Knowing what workflows are and how they function, we can now look at the application of workflows in general and more specific in our domain. First, we will look at workflow functions in the healthcare domain and the strengths and weaknesses of workflows in general. Based on these two we can then determine the potential of workflow systems in our residential care home domain. In this examination we will look at the "default" circumstances (normal administrative tasks) as well as the setting of the multidisciplinary meeting. After determining the workflow potential, we will end by looking at some of the related initiatives of workflow application within the general healthcare domain.

3.4.1 Workflow Functions

Because workflows are, at the moment of writing, barely used within our domain of residential care homes, we have also looked at the hospital healthcare environment to find functions which workflows could or already fulfil. According to Song et al. [23], workflows can be used as user interface (UI) guidance systems, medical workflow monitors and critiques, reminders and/or alerter(s), healthcare quality and financial review generator and as support tools for clinical decisions. We will now shortly discuss these options:

UI Guidance System Dwivedi et al. [5] state that workflow technology in combination with the rapid growth of the internet could be used to create healthcare solutions which revolutionize the way we handle medical data. The increased usage of internet directly relates to the growth of applications which distributes electronic healthcare data of a client among different healthcare specialists. Because the amount of information that is involved grows rapidly there is need for some sort of "guiding" entity to determine which information should be available at which location at a certain time. UI guidance systems support single-user sessions and focus on using advanced UI techniques. Using these techniques it should be possible to create a system which dynamically switches its context to the right client and retrieves or summarizes certain information automatically.

Workflow Monitor and Critique Workflow monitoring and criticism originates from the idea that every procedure has a certain default method (or template). As the healthcare professional progresses with treatment and inputs his method, the system can than "criticize" the professional about the deviations he took from the original method.

Reminders and/or Alerters Workflows can also be used as reminders or alerters, as used in scheduling and planning environments. It is safe to assume that healthcare environments are largely human-driven and therefore such system might be highly valuable. Using workflows as alerters can be seen as workflows instances that are being launched as a response to a certain predefined condition. For instance, in case a certain result of a medical test is entered, the person who ordered the test is notified immediately.

Review Generator Healthcare quality and financial review generators are used to analyze past workflow events, analyzing log files. Based on these logs it may be possible to improve procedures. This will increase the efficiency on which (especially administrative) tasks can be performed and as result may lower the overall costs.

Clinical Support Tool Within a clinical support tool, the workflow system is used to provide advice to physicians by referencing detailed clinical guidelines in a knowledge base.

Although our environment is different, we reckon that certain functions, like UI guidance, reminders and/or alerters and review generators could be used in our environment also. We will return to these choices in our conclusion.

3.4.2 Strengths and Weaknesses of Workflows

Before concluding where and how to use workflow systems as an addition, we must determine the strengths and weaknesses of workflow systems. Although we cannot fully determine where problems will occur, we should be able to make some reservations about the usage of workflow systems. This section describes some of the general workflow issues we encountered during our research: flexibility and freedom of modelling.

Flexibility A few issues should be taken into account while working with workflow systems. First of all workflow systems are traditionally designed to work in a static environment. This means that except from the changes in workflow models, the actual execution of a model as such remains constant. If a change in a certain process occurs, we either have to create a new workflow model, or restart all workflow instances with the new process parameters. (we cannot change the working of a process during the execution of an instance.) The introduction of this kind of flexibility is difficult, but in some situations highly useful.

Imagine a group of algorithms, each optimized for specific situation. Let's assume we cannot determine which type of situation will occur prior to the workflow launch. So we are unable to determine which algorithm should be present in the process itself.

Worklets [1] are one of the possible solutions for this problem. Worklets can be seen as little pieces of abstract code which, at run time, determine which implementation should be executed. Whether there is a workable solution or not, one should keep in mind that workflow systems are, at this moment, still static systems.

Modelling The second issue is with regard to modelling. Modelling allows the user to create workflow models that react to his specific need. This should be considered strength of the workflow systems. What is the point of using a workflow system if we cannot suit it to our needs? However, this level of freedom also introduces a new weakness especially within the domain of residential care homes. The procedures within the home care domain are generally static on a broad level, but subjected to change on a minor level constantly. (e.g. the insurance company wants some additional information or prefers it in a different format.) If a procedure changes, it means that the workflow instances currently executing should be updated as well. The creation or modification of a workflow model is generally considered difficult and is usually performed by business experts. In short, the freedom of modelling allows workflows to be adjusted to a certain situation. However, if that situation is subjected to perpetual change, workflow modelling becomes challenging, if not by complexity than as time consuming activity.

3.4.3 Workflow Potential

Now we will get more formal and determine where workflow systems could be employed best (also known as the workflow potential). We will split our question (where to apply workflow systems) into separate categories. First of all, we will look at the administrative field. This is where only administrative tasks within our residential care home domain take place. These tasks include the admittance of a patient, the communication with healthcare insurance companies, etc. The second field we will discuss is that of care provision. The care providing tasks are focused on the actual treatment of clients. Although this is not our main focus for this research, we do wish to address this topic to provide some insight. The last topic that will be reviewed is that of multidisciplinary meetings. For the basis of this section we have chosen to use the function list from the previous section and try to determine the potential for each of the found functions.

Administrative Field

The administrative field is the first domain which comes to mind with regards to automation using workflow systems. The administrative field contains most of the tasks related to the paperwork surrounding a client. This ranges from client details to the reports that are sent to the client's insurance company. We will now look at the functions from a hypothetical point of view. We can select certain functions, but without the actual testing and evaluation of these functions, we cannot conclude which is the best. The following section is the basis for the prototype we will discuss in the next chapter.

From what we have learned so far we find that UI guidance and review generators are the best functions that apply to this domain. Because healthcare organisations tend to grow larger and larger, an intelligent system that can guide a user through one or several applications can be considered desirable. Using a guidance system, the efficiency of the tasks performed can be increased while the level of expertise required to use an application could be reduced. This increase of efficiency could be obtained by the fact that users are

"coached" while doing a task (e.g. in case of distractions) and so wrong actions being taken while completing a task is prevented.

Review generators can be used to determine the efficiency of the overall process and outline the problem areas of certain activities. In general we can assume that administrative tasks are repeated often and thus are prime candidates for automation (and task optimisation). By generating statistics with regards to task duration and average transfer times (between users) administrative tasks can be improved.

Care Provision

Care providing tasks are more complex than administrative tasks. Also, care providers generally use different methods to help a client. So, the role that a workflow system needs to fulfil will be different as well. For instance, we expect that the function of UI guider will be less important. Although arguably we can say that guidance is even more important in care providing tasks, we assume that the task performance and efficiency is highly related to the expertise of the healthcare professional providing the care.

For this particular section, we opt to use the previously mentioned reminder and alerter functions, as well as workflow monitor and critique.

First of all, the reminder and alerter function was used to support planning and time scheduling of healthcare provision. Today in most healthcare environment timing and planning issues are becoming more important day-by-day. We reckon that planning and time scheduling problems are the most important within this particular domain (for instance in comparison to the administrative field). Therefore a workflow system should be used as a reminder / alerter support tool.

The second function a workflow could fulfil is that of workflow monitor and critique. Healthcare providing tasks usually have some framework on "how to act" in certain situations. Although these guidelines are not strict, they are supposed to be followed as much as possible. In this setting, having a workflow system which monitors actions and provides "critique" (we would prefer the term "advise") to healthcare employees can be considered a valued addition.

Multidisciplinary Meetings

The last type of setting in which we would like to investigate workflow potential is that of the multidisciplinary meeting. Based on the interview conducted at the *Atlant Zorggroep* we learned that multidisciplinary environments in residential care homes are different from multidisciplinary meetings in the hospital environment. Therefore the workflow potential also differs from that of the regular healthcare environment. Pinelle and Gutwin [20] report that within multidisciplinary home care environments there are generally five different areas which should be supported to improve collaboration within the team. These are scheduling, information dissemination, information retrieval, short-term treatment coordination and long-term treatment planning. Pinelle and Gutwin also provide some recommendations with regards to supporting these areas. Most interesting of these recommendations, although

being slightly outdated, is the part on information handling in general. Even in 2002 it was reckoned that information dissemination and retrieval is a potentially time consuming and tedious job. Providing a (workflow) system which automatically guides a user to the right pieces of information might be valuable.

However, we also learned that multidisciplinary meetings in the residential care home environment are generally social processes which comprise little administrative activities. Mind, the care expert must be able to access the client's data. Based on this assumption we have tried to develop certain functions which could be implemented by some form of automated system. The problem with our approach however was that it involves the use of ICT systems to provide such assistance. We know from the interview with André Middelkoop and conversations with other healthcare employees, that the use of any system is nearly always undesired during these meetings.

As input for each MDM a list of issues is cross-checked by the primary caretaker with the client to determine the results. The result will be the input for the next MDM. We would like to see if a system can be created which produces a good checklist for all issues to be discussed with the client. Along with just showing the (un)resolved issues there should be a possibility to quickly report the found results to their respective disciplines so that specialists can prepare for the MDM, knowing the status of their issues.

3.4.4 Related Workflow Systems

This section covers various sources of literature that were found related to this topic. The described problem in this thesis however, is rather new and yet still unexplored. This results in a limited amount of (academic) literature that could be found with regards to this topic. This section provides an overview of the found results but should not be considered complete in any way.

Soarian Clinicals The idea of using workflow systems in healthcare environments is not new. Hospitals use workflow systems for various reasons, ranging from medical treatment plans to the billing of a patient for healthcare services. Although the environment is different from ours in this thesis, we are able to draw some knowledge from this domain. Siemens offers a system, called Soarian Clinicals [6, 9], which uses workflow engines to analyse situations, determine new plans and take actions in the healthcare environment. Soarian Clinicals is advertised as the healthcare system which does more than "just" deliver information.

It seems (unconfirmed due to lack of literature) Soarian uses the workflow engine to respond to certain scenarios. The workflow models are launched as soon as certain conditions have been met or a specific action is taken within the healthcare system. Figure 3.6 shows the main components of Soarian. Recalling the overview of a complete workflow system, some similarities can be seen. First of all, the workflow engine uses a process definition (e.g. workflow model) as basis and runs on top of some database system. Based on actions or events coming from the healthcare information system certain workflow instances are launched and during the execution data can be retrieved from the healthcare information system. The

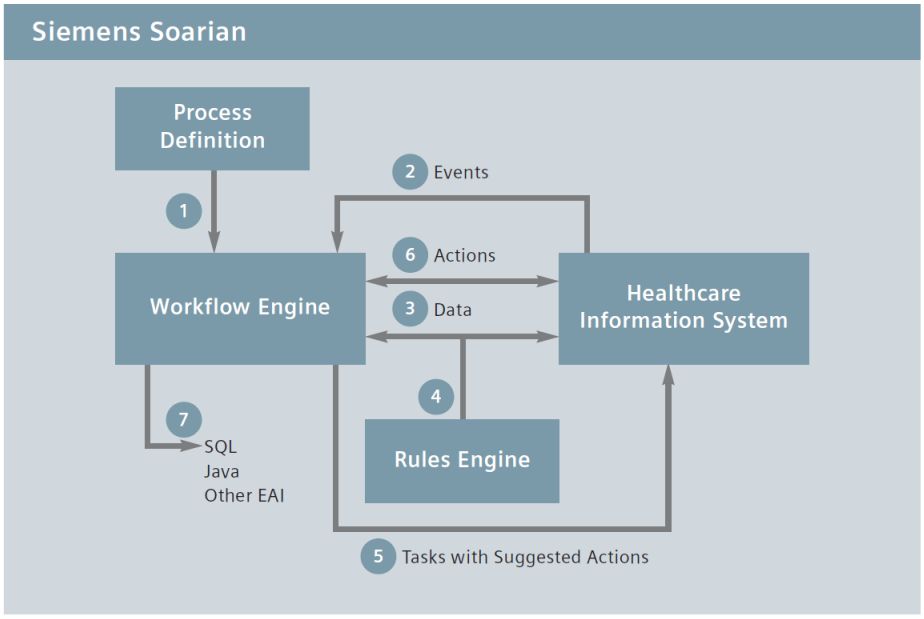

Figure 3.6: An overview of the Soarian system, obtained from [6]

exact function of the rules engine is unknown to us, however we may assume that access to certain information is restricted and a rules engine might be used to determine access levels. Finally we see that during the execution or completion of a workflow instance tasks with suggested actions are send to the healthcare information system, the main interface for the user. We are confident that such a guiding approach of workflow usage is also applicable in our environment and therefore we will use Soarian Clinicals as basis for our prototype in creating a "guiding" system.

RetroGuide One of the more recent projects focusing on using workflow technology in healthcare is called RetroGuide. RetroGuide is an ongoing project in biomedical informatics which focuses on the application of information technology within a medical setting. One of these applications involves the usage of workflows. RetroGuide is a suite of tools belonging to larger projects by the name of FlowGuide and HealthFlow. These systems were designed to execute process models and to experiment with the use of workflow technology in the context of (among others) clinical decision support. It has been developed at Intermountain Healthcare [10] and Department of Biomedical Informatics at University of Utah in Salt Lake City. At the moment, no academic papers related to RetroGuide, HealthFlow or FlowGuide could be found, but the overall RetroGuide system was evaluated in a dissertation by Huser [12][4]. For this evaluation he used informatics users with small to moderate analytical experience and the goal was to determine if the barrier of analyzing data stored in a large database (Enterprise Data Warehouse) would become lower by using workflows

[4]The complete dissertation could not be located in the online archives, only one chapter with regards to RetroGuide could be found

rather than using the default database utilities (like the SQL language). It was chosen to compare RetroGuide with standard SQL-based analytical technology (no details are known of this technology) and focus of the evaluation was set at the method of inputting the queries into the system (i.e. how should the questions be asked). The participants were asked to complete certain assignments which were then evaluated by two different score-markers. The total group size of this evaluation was 18 people. The complete statistical results of this evaluation shows a significant difference between the two different methods, showing that the workflow modelling approach used in RetroGuide was better suited than the default SQL analytical approach. Note however that one of the main goals of RetroGuide is to support clinical decisions, which is rather uncommon within our specific environment.

3.5 Summary and Conclusion

In this chapter the domain of workflow systems was explored. We have seen scientific workflows and business workflows and have concluded that the latter would be best suited for our domain. We moved on to the workings of workflow systems in general and looked at workflow models (the instructions to a workflow system) and workflow instances (instantiated copies of models). A distinction was made between different process types and we concluded that hybrid processes are best suited for this particular domain. Choosing for hybrid processes meant that we have to create a method of making these processes accessible for humans. This means that the process definitions should include instructions which can be understood by a human, rather than a compiler. We will look at that in more detail in the prototype chapter.

We have seen the complete overview of a workflow system, looking at the different interfaces in which communication between a third system and a workflow system could be established. We have seen how processes can issue tasks (workitems) and what they are made of and how they are stored within the workflow system.

YAWL was introduced as our workflow system of choice. We have chosen the YAWL workflow system because of its academic basis and the fact that it nearly matched all our pre-established requirements.

The found workflow functions (ui guidance, workflow monitor and critique, reminders and/or alerters, review generator, clinical support tool) and known strengths and weaknesses of workflows (the lack of flexibility and the freedom of modelling) led us to draw a conclusion about the workflow potential within our domain. We concluded that administrative tasks would be best supported by workflow systems acting as guides through their respective systems. Also the review generator functionality seems promising as it can increase productivity and efficiency. For the care providing tasks we believe that role of workflow monitor and critique was best suited, also the reminder and alerter function seems promising with regards to scheduling. The multidisciplinary meeting, while (yet) unsuited to be supported by ICT in general, could benefit from the guidance functionality as well as the report generation. Looking at the goal of a multidisciplinary meeting (to evaluate current processes to keep a certain standard) clinical support functions could also be considered. A

Table 3.1: Function Potential For Specific Domain

	UG	MC	RA	RG	CST
Administrative Tasks	✔			✔	
Care Providing Tasks		✔	✔		
Multidisciplinary Meeting	✔			✔	(✔)

summarized overview can be found in Table 3.1. (UG = Ui Guidance, MC = Monitor and Critique, RA = Reminder and/or Alerter, RG = Review Generator and CST = Clinical Support Tool)

Thanks to the literature on Soarian Clinicals we now have a better understanding on the relation between healthcare information systems and workflow engines. We will base our prototype on the described layout, keeping the workflow engine and information system as two separate systems but able to communicate if required, as this seems the most obvious and logical. RetroGuide made us aware of the importance of the application of workflow technology in our domain.

In conclusion, we see that there are several opportunities that could be explored. Because we are unable to create prototypes for all different possibilities, we have chosen to limit ourselves to the administrative domain. Although the care providing domain also has some interesting topics, we reckon that the amount of additional research involved would be too much for this thesis. The multidisciplinary meeting was found to be uninteresting to support with ICT by the healthcare environment itself and thus was not a logical choice to proceed with for this research.

During the creation of our prototype we will focus on the guidance function, meaning that we will try and create a prototype that will guide a user through an administrative process. More about this prototype in the next chapter.

Chapter 4

Prototyping

To demonstrate how workflows could be put to use in a healthcare environment, two different prototypes were designed: the Topicus Task Manager (TTM) and the Snippet Factory (SF). In this chapter we will discuss the global idea behind each of the prototypes. Due to time constraints, we could only implement and test the Topicus Task Manager.

4.1 Topicus Task Manager

Our first prototype, the Topicus Task Manager, was created as a system to demonstrate a possible use of workflow technology combined with already existing healthcare software. The role of this workflow addition to an existing system (in our case TisWeb, which is discussed later) should be that of issuer and monitor of "tasks". Before we proceed on exploring the prototype in more detail we need to redefine the concept of tasks.

Tasks Within this thesis we have used several descriptions for a task but none of them was completely appropriate for this chapter. In this section we will rectify this problem. The trivial meaning of the word task, as found in any dictionary [1] is not sufficient and needs to be extended.

A *task* is still something that needs to be done, an objective that needs to be accomplished or a job that needs fulfilling. However when we talk about (human) tasks, we always assume that there is some additional context present to clarify the task or tell us how to reach the objectives set for a task. "Our" task has no such external provision of context, but contains its entire "context" internally. Consider a task to be a collection of facts and statements, all related to the accomplishment of the overall job it is representing. For example, the name of a task ("Taking out the trash") does not inform the user of any method of accomplishing this task. To reason with tasks individually we have constructed a method of combining all the required information into one container. The task thus forms the container carrying all relevant facts about the job it describes. (We will see this in the implementation of the Task class in our prototype, see more in section 4.1.2.)

[1] "A piece of work assigned or done as part of one's duties."

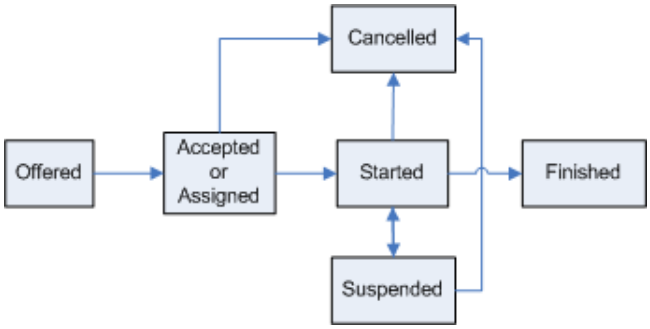

Figure 4.1: The different states of a Task.

A task will contain a name, some human-readable description, a reference to the client it is related to (if any), the moment it was created and much more. It also includes a WorkItemRecord (chapter 3), the basic component the workflow engine uses to keep track of progress and uses in its communication protocols. These WorkItemRecords are used as communication tokens with the workflow engine. The workflow engine itself retains a certain state for these WorkItemRecords and thus implicitly for our tasks. There are different states in which a task can be: *offered*, *accepted*, *started*, *suspended*, *finished* and *cancelled*. Figure 4.1 shows how these states can be obtained and how they related to each other. The YAWL workflow engine does not contain the *finished* state, as it does not keep track of previously finished tasks: it simply deletes the WorkItemRecord.

Because we want to track *tasks*, we need to incorporate some more information about a task within the task itself. Originally we wanted to use the WorkItemRecords of the workflow engine itself, but this method was rather limited and unpleasant to use. (E.g. why bother transmitting a complete task context to a workflow engine when the engine itself couldn't care less about the details?) We therefore opted to create our own definition for *task* which includes all relevant details.

4.1.1 Goal

As tasks in normal life are highly variable, we have chosen to restrict ourselves for this prototype to administrative tasks only, as described in subsection 3.4.3. We allow the modelling of administrative processes, as long as they can be performed by the related system. For the sake of simplicity we have chosen to only allow one system to be connected to our prototype at any given time, although in theory it could support several. We will come back to this later.

Workflows in general are used as a method of "guidance". The method of how workflows are implemented within any workflow-using system can vary highly. It could be that the workflow engine is connected to the core of the system so that the user is completely oblivious to its presence. Otherwise it could be that the user sees glimpses of the workflow model, without being bothered with the actual implementation of how this model is enacted.

To determine if a rather direct interaction between the user and the workflow system is

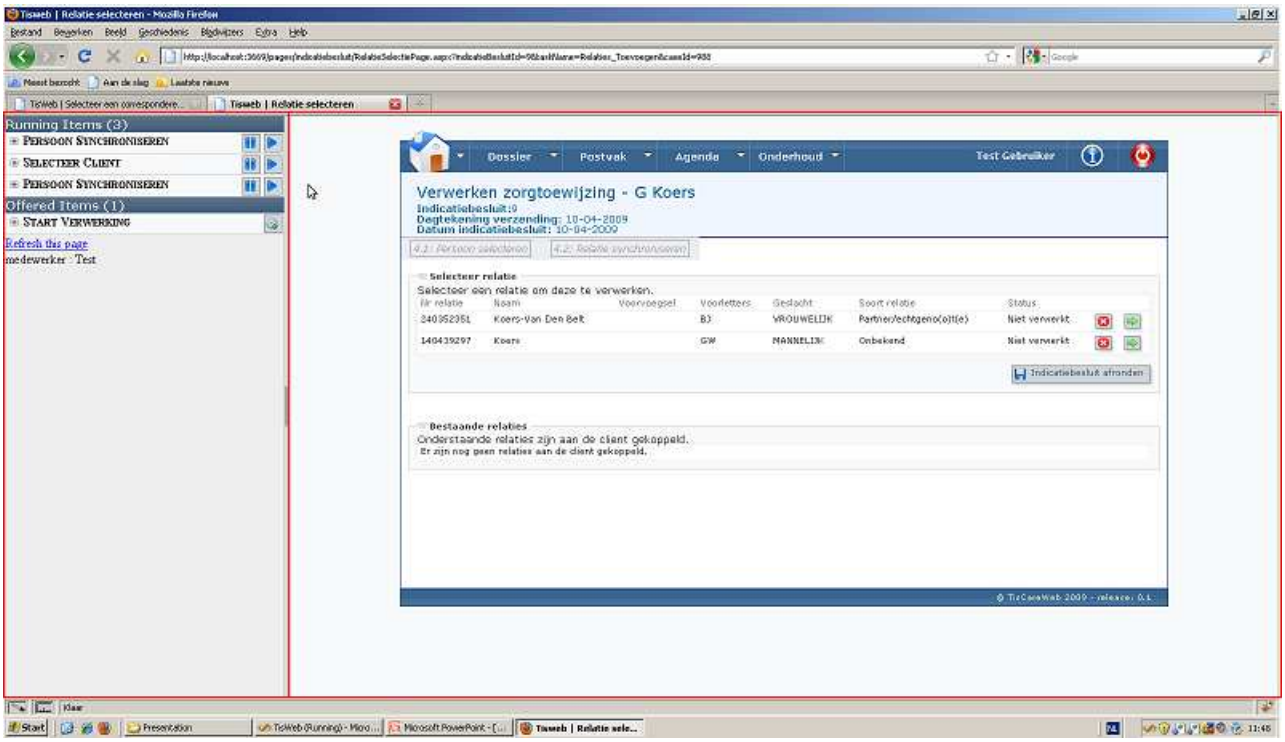

Figure 4.2: The dual-framed interface, marked with a red border.

desirable, a proof of concept was created which allows users to assign tasks to themselves and execute them. As shown in Figure 4.2, a dual framed interface was created which showed the workflow component on the left and the healthcare system (TisWeb) on the right. The user is immediately directed to the *offered* task section, where all tasks available to him (based on either his username or role within the organisation) are displayed. This section can be seen as a large shared pool of tasks, where all users with certain characteristics can obtain tasks from. By a simple mouse-click the user then assigns a task to himself. Once the user assigns himself a task, this particular task is removed from the shared pool so no other user can assign this task to himself. A possible scenario:

User U gets the assignment to complete all tasks related to client C. U loads the web-application which he normally uses during his daily routines. A task frame is shown at the left side of his screen. Within this screen he notices different tasks offered to him (and his colleagues) and determines which tasks are of the highest priority. He then manually assigns the tasks to himself by a simple mouse click on the chosen tasks and notices how these tasks appear in his own private "Running items" list. At this moment the task is officially assigned to U and U can start the task by clicking the play-button in the frame. The regular application process will start at the correct page and U can do his work. When the user is ready, the task manager registers the completion of the task and offers the proceeding task (based on the workflow model) to the system. This task can either be placed in the shared pool, or directly

assigned to U, depending on the workflow mode configuration[2]

Limitations

During the development of any prototype, boundaries must be set to limit the costs (time, money, etc). For our prototype we have only considered a small scenario from the complete system. This scenario is called *Indicatiebesluit Uploaden*. This is the processing of a large data file, received by a healthcare organisation which includes the details of new clients (or updated details of existing clients). We will now look at this scenario in more detail.

Scenario As we were unable to randomly generate tasks for testing, we have chosen to create a scenario based on an existing and known procedure within our domain. This scenario is called *Indicatiebesluit Uploaden* (Upload a file with client indications) and exists of the following steps:

- Upload the *Indicatiebesluit* file.
- Start the synchronization process.
- Select a client to be synchronized.
- Synchronize the clients details with the existing information in the database or create a new client entry.
- Add any known family relatives of the client by adding the relatives to the database.

Without going into too much detail, all the steps mentioned above can be performed by either a single mouse-click, or typing a bit of text and then clicking a button. The challenge for our prototype is to detect whenever a certain task is completed. To keep our prototype simple, we trust the user successfully completes a task. For instance we will not verify if all organisation-specific conditions are met when the user claims he completed the task, by example "all fields are filled").

The prototype instantiates a workflow model, based on the scenario that was created. It will then offer tasks to users and once a user accepts a task, tracks the progress of this task until it is finished. During this tracking, it allows the user to quickly swap between tasks by storing the context and providing a direct link between the pages in the regular system.

4.1.2 Methodology

For the implementation of the Topicus Task Manager we used a pre-build code "template" which was created by Topicus. This specific "template" uses Spring [24], a method of quickly

[2]The YAWL engine allows workflow models to be annotated with additional information regarding task assignment. E.g. it is possible to directly assign a task automatically to a certain person / role / etc. instead of offering it to the shared pool.

swapping classes at run-time, and Hibernate [13], a persistency layer for java on top of certain databases. The use of this template was highly recommended by the developers of Topicus because the configuration of both systems can be quite complex for beginning users and "we would need it in the end" anyhow. In short, this template allowed us to immediately start development of the prototype, rather than spend a lot of time reading instructions on how to configure basic database access. As we are developing a web-application, the layout needed to be in the form of a webpage. We were advised to use Wicket [7] to be able to connect the Java programming parts to the HTML front-end. Wickets allows the html definition of certain fields (hyperlinks, tables cells) of which the content can be manipulated using Java.

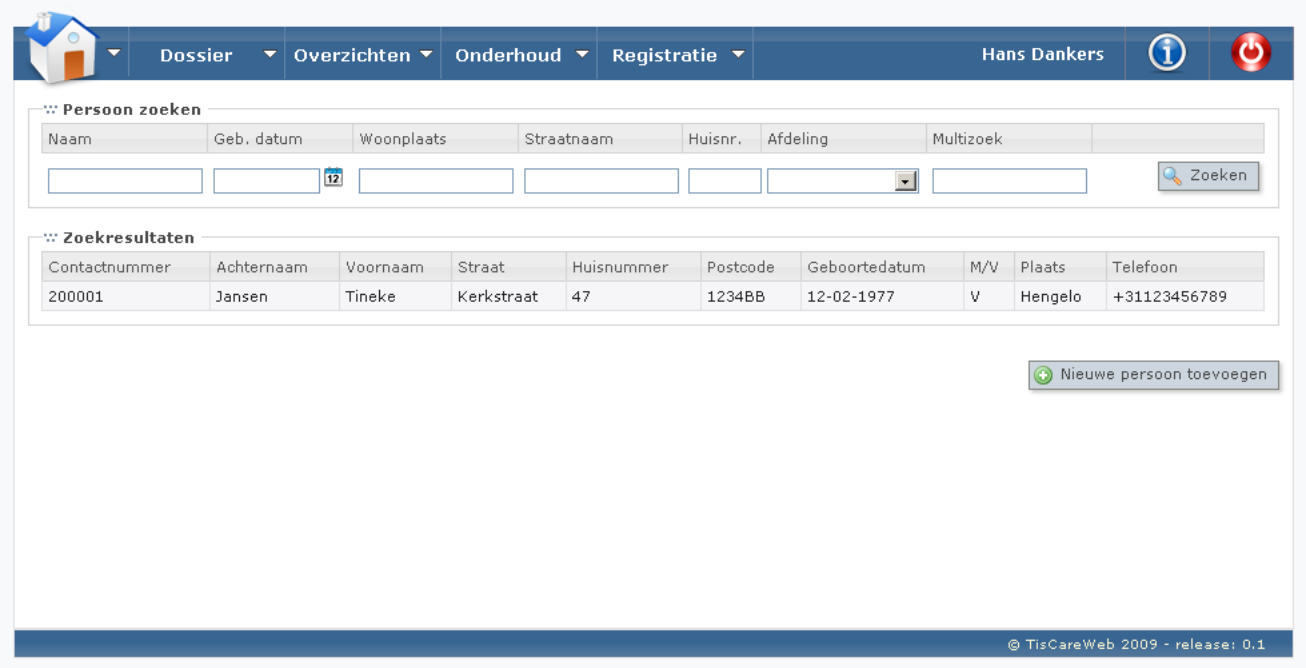

Figure 4.3: The TisCare Web Interface.

The connecting system on which we created the prototype is called TisCare Web, or TisWeb in short. TisWeb is a web-application which computerizes the medical archives of a healthcare organisation. It provides a central access point for users to log on. The system takes care of permissions, allowing and restricting user access for specific parts of a client's file. The goal for this system is that both the client and healthcare employees can log on one central system where all the information is stored. TisWeb was originally designed to be (amongst others) a new front-end for a larger database system which had become too complicated for end-users. TisWeb is at the moment of writing in the final stages of development and proved to be a highly suitable platform on which the proof of concept could be developed. For obvious reasons we are not allowed to discuss the inner workings of TisWeb in more detail than we will in this thesis. An example screenshot of TisWeb can be found in Figure 4.3.

The remainder of this section clarifies more of the inner workings of the prototype. We

will do this by looking at the structure of the prototype, analyzing the components by their roles, looking at the method of communication and how they all interact. We will first look at the overall picture: what are the main concepts in this prototype and how do they function. Next we will look at some of the concepts and components individually.

From The Top

From the highest level this prototype is a separate web application. A schematic overview can be found in Figure 4.4. The prototype consists of actually not one, but multiple web services combined. The YAWL workflow engine itself is a web service, however we have chosen to incorporate it into our prototype. This was done because having YAWL as a separate web service would require an additional layer of communication (namely XML / SOAP). In short, the YAWL engine was included in the prototype's source code.

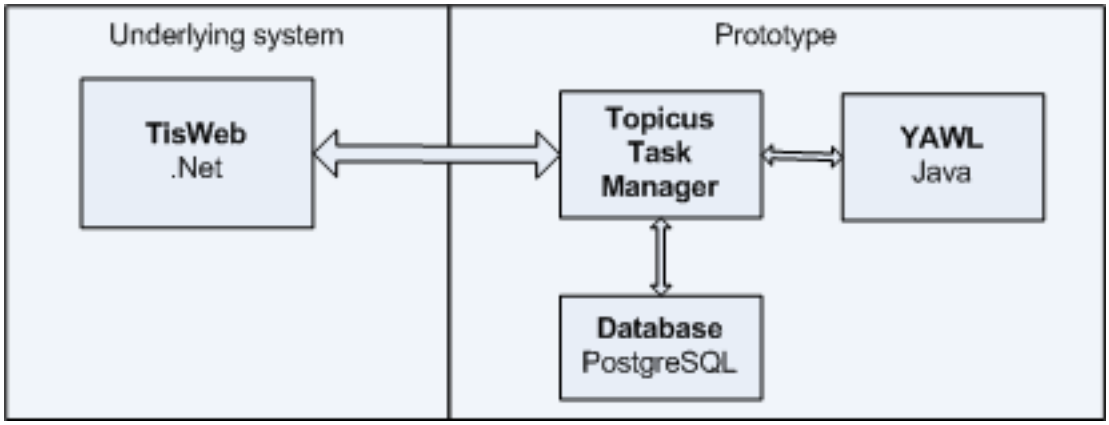

Figure 4.4: A complete overview of the connected systems.

The first thing that should be noted is that besides the prototype, the existing system and the YAWL workflow engine, an additional database server is shown. This database is required because we need to store tasks as described earlier. The workflow engine itself works only with WorkItemRecords and these are insufficient to store all the data. As the workflow engine itself does not store any task-related information once a task is completed there is no method of keeping track of tasks that already have been finished. Because the healthcare domain is obligated to keep track of every action it takes with regards to a client, it makes sense that tasks in their totality are stored for future reference.

Zooming In

Figure 4.5 shows a more detailed overview of what is going on inside the Topicus Task Manager. We distinguished four different items of our prototype worth looking at in more detail.

The first item we would like to discuss is the *Task* implementation class. As we have seen before, tasks are containers of facts. They contain the description of how the task should be

accomplished in case it is a human driven task, a piece of code in case the task is automated, a set of identifiers to be able to distinguish this particular task from other similar tasks, etc. The Task class also is responsible for generating the correct URL for the underlying system to be able to quickly navigate to and between different tasks. It can be compared to using bookmarks in most internet browsers. The only difference between these bookmarks and our system is that the URL's generated by the Task class are context-specific for a certain client. Because we want to provide some additional means of interacting between different users who work on a task, we have also included a comment-attribute to the task, which is propagated between different succeeding tasks. This means that, with regards to our scenario, a user can enter some comment on the first step, and another user can read this comment at the last.

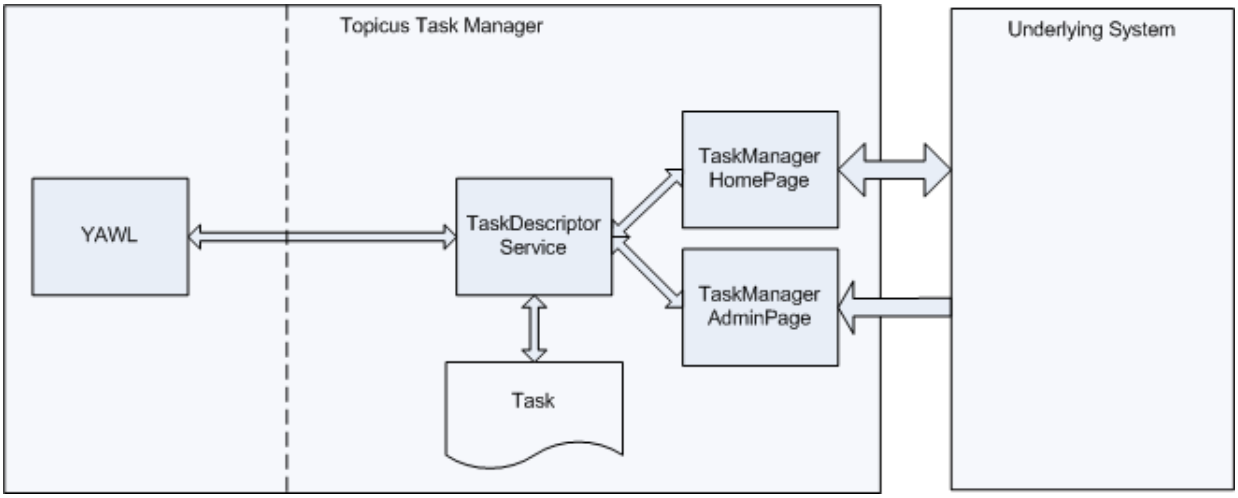

Figure 4.5: An overview of the connected programming components.

The second item is called the *TaskDescriptorService*. The TaskDescriptorService was created to handle all possible manipulations of tasks. It is the core of the prototype. It receives instructions from the TaskManagerHomePage and TaskManagerAdminPage and translates this to workflow engine instructions. It also stores and retrieves task information from the database. Its key function within our prototype is to ensure that the mapping of WorkItemRecord in the workflow engine to an annotated task stored in the database is done correctly. This is relatively hard as the service does not know the workflow model. This means that at the moment one task is completed "some other new" task is started, but the service has no idea which task to expect. We have resolved this issue by checking for each WorkItemRecord that comes in from the workflow engine if a database entry corresponding to that record exists. Because of this continuously rechecking (in programming terms a costly step), we reckon that the current setup is not ideal. A better solution would be to make the workflow model known to the prototype as well as the workflow engine. The introduction of the workflow model into the prototype means that it is always known what task to expect

and we can prepare for the arrival of this task accordingly [3]. Additionally we can use such an incorporated model to include better graphical aids to display overall progress.

Figure 4.6: The TaskManagerHomePage.

The third item is called the *TaskManagerHomePage*. The TaskManagerHomePage is the actual list of tasks that is displayed in the side-frame. Figure 4.6 shows the frame in which all task details are listed. As we can see there is a distinction between the running items, where all the accepted and assigned tasks are stored and the offered items list. The latter is the list containing all the offered tasks to this particular user (either by role or username). The lists are sorted by task name and each task can be unfolded to display the task-context. As shown the comment section is also included, but due to last minute change in the prototype there is no method of actually filling it with comments for a particular task. We have chosen to list all available tasks on basis of the task name, because this is precisely how the workflow engine itself processes WorkItemRecords. In retrospect (and based on the evaluation which we will discuss later on) this decision might not have been the best choice. We will return to this discussion later.

The final item is called the *TaskManagerAdminPage*. The TaskManagerAdminPage was created to accept any incoming communication from the underlying system. It needed to be fast and should not contain difficult bits of programming. In other words, this page is "simple and dumb". The TaskManagerAdminPage analyzes the parameters that are passed through the URL and knows three different commands; *launchCase*, *finishTask* and *updateTask*. These commands represent in order the launching of a new workflow instance (e.g. the indication file has been uploaded), the successful completion of a task or the updating of task information with details from the connected system. Especially the *updateTask* command became a necessity for the pre and post condition problem which we will discuss in the second prototype. We can describe this problem as followed: To generate an URL corresponding

[3]This does not eliminate the problem of pre and post conditions on tasks which we will discuss at the second prototype

to the right page and context to fulfil a task, certain detailed information from the related system must be known (e.g. a number which identifies the client in this particular system). The first step of our scenario involves the selection of a person from a list, e.g. we are not sure which client we are going to process and therefore have no such identification number. At the moment that identification number becomes known (e.g. the user selected the right person from the list), the related systems sends an *updateTask* command to the task manager, which then (by means of the TaskDescriptorService) updates the task context for any tasks which require that particular identification number to be part of that context. We will return to this issue in a more general form in the next section.

4.1.3 Walkthrough

For a better understanding of the mechanics of the prototype, we have chosen to create a walkthrough which illustrates the functions the prototype offers. This section describes the interaction between the prototype and some other system. For the walkthrough we have used the TisWeb system we described earlier. There will be no new information given about the prototype except for design issues. This walkthrough is based on the scenario described earlier in this chapter. The figures related to this walkthrough can be found in Appendix A.

At the moment the system is loaded, the task manager is idle and resides in a hidden frame in the browser (Figure A.1). By clicking on the bar *Taken* on the left of the screen the task manager gets pulled out of hiding and now takes approx. $\frac{1}{5}$ of the complete screen (Figure A.2). The size of this frame can be adjusted but for this prototype we gave the frame a fixed size. Once the system is loaded and made visible, the default state is that there are no tasks visible. This is because no workflows are initiated and thus no tasks are offered to the users. In case we are resuming a previous session by the user (e.g. the browser crashed or a new day of work started) the lists would already be filled with the user's unresolved tasks. For this prototype, there is also some debug information present and a refresh button (see also Figure 4.6). The refresh button was implemented to reload the task lists because the updating of this list does not occur in real time and is not scripted to be performed at a certain interval. The lines below the refresh button are the parameters sent by the related system to the task manager. We have chosen to display these lines in order to demonstrate how the URL based communication takes place between the two systems. One parameter that is always sent to the task manager is *Medewerker* (Employee) which identifies the user currently logged on the related system (and whose tasks are displayed). At this moment both the task manager and related system are ready, but appear idle to the user.

After a *Indicatiebesluit bestand* (a file containing several *Indicatiebesluiten*) is uploaded, workflow instances are created for each of the clients found in the file. In our scenario this means that there are four different workflow instances created and thus four tasks added to the shared pool (Figure A.3).

From here the user either gets a task assigned or can accept a task himself (e.g. clicking on the configuration or cog-wheel icon at the end of the line). The task is moved from the shared pool to the user's private pool (Figure A.4). An accepted or running task has two buttons: a pause button and a play button (as found in popular icon sets). The pause

button in our prototype is used to freeze the execution of a task. Freezing in this sense means that the total amount of time required for this task stops counting. Our prototype does not perform any other action with this button.

The second button, the play button, is also known as the navigation button. Clicking on this button will load the correct context of a task and navigate to the right page within the underlying system where the task needs to be performed. The URL which is required for this navigation is generated by the task manager on basis of previously obtained task data (remember the section about *updateTask*). At this moment the user is allowed to navigate freely around the system. It is possible to unfold a task which will display the task's details. An example of the system in use, where one of the tasks is unfolded can be found in Figure A.5.

The only issue still to be described is the completion of a task. Task completion originates from the underlying system, e.g. only that system knows when a certain task is complete. This means that the underlying system must be aware of what kind of tasks exists and what conditions must be met for a task to be completed. Our prototype resolves this issue by sending the task name to the underlying system. Certain key points (like accept or apply buttons) were adjusted to send out a task completion signal to the task manager.

4.1.4 Issues and Resolutions

As with any development we ran into certain issues for which no perfect solution could be found. This section describes some of the more major issues that rose during development.

Task Retrieval The first problem we encountered was that of task retrieval. The decision was made that tasks would be annotated and stored separately. The YAWL workflow engine could not be used to store annotated tasks because it was unable to handle these more complex data structures. A mapping between the database and the workflow model had to be established.

The workflow engine is queried on regular interval for all tasks contained within a certain state. Each state within the YAWL engine is internally represented by a queue containing WorkItemRecords. E.g. the *OfferedQueue* contains WorkItemRecords with the state of *offered*, etc. These queues can be queried by means of the YAWL API. At the moment this query is completed for each of the found WorkItemRecord (they are no tasks yet!) a task is constructed and stored in the database. However, at the moment the system is restarted, the information is lost on whether a task is new or already exists. We therefore query the database to see if an entry can be found matching the description of the task. In theory, it would be best to store the WorkItemRecord itself in the database, but this cannot be accomplished as WorkItemRecord are continually updated and therefore change. We have chosen to use a combination of attributes (task name and case id) to form a unique identifier. Based on this identifier we can now query the database to find the right task. We realize however that our method of creating such a unique identifier is problematic and would most likely fail in larger and complex cases. A better method of providing unique identifiers is

to include something that is by definition unique on its own, like the clients social security number.

Workflow Model Complexity The YAWL model editor allows for the creation of complex workflows. AND and OR conditionals have been introduced to create additional freedom for models. All this additional freedom also provides complexity and is quite undesired for the prototype. As we have seen in task retrieval we have currently no method of finding unique tasks based on pieces of information. This results in a restriction we have to impose on the workflow model itself. We have chosen for our prototype not to consider workflow models which contain a certain task twice (or more). An additional problem arose at conditionals. It is possible to create a workflow model which contains an OR conditional. This means that e.g. from a selection of two tasks only one has to be completed. Since our prototype has no knowledge of the workflow model, it is impossible to determine when such a situation occurs. We therefore chose not to include parallel workflow models in our prototype. (In fact, our scenario was chosen because it did not contain such events.)

4.1.5 Summary and Conclusion

For our prototype we have tried to create an interface to integrate the concept of tasks within an existing system. We have chosen to create a prototype that is omni-present and at all times provides the user with a list of tasks that need completion. We have chosen to make a distinction between *Offered* and *Running* tasks. Offered tasks are tasks that exist within a shared pool. This pool is shared between members of a certain role within the organisation. Running tasks are tasks that have been assigned to an individual user. This assignment can either be done by the user himself (e.g. accepting a task) or by a supervisor (e.g. assigning a task). At the moment the tasks are assigned, the duration and launch time (among others) of a task are recorded for statistics. The task manager keeps track of task progression by sending a complete-task signal towards the workflow engine as soon as the system registers an action which indicates its completion. For our prototype these completing events are hard-coded within the related system. We also have chosen not to perform any action with regards to the derived statistics.

We have chosen to display tasks by their name. This allows for easy sorting of tasks and prioritizing by a user (e.g. the user sees the names of the tasks and knows which to one do first). The task manager also provides additional information with regards to a task. During the design of a task some additional information, such as task description, can be added to clarify the task. There is also a possibility of adding comments to a certain task. This comment is then propagated between the tasks of a certain workflow instance.

Finally the task manager can be used as an assistant for easy task switching. Because normally the underlying system would not keep track of its context[4] the system saves this context within the to-be generated URL. This means that once the corresponding task is selected within the task manager, a direct URL can be generated which takes the user to

[4]In certain situations TisWeb retains the context, though this is more exceptional than default

the page within the related system with the full context. This allows users to quickly switch between tasks without having to worry about loading the correct context into the system.

We have tried to create a prototype that is easy to understand, quick and always accessible. This means that the information presented within the prototype should be clear and direct and the options limited to those that are strictly required for a good functioning. The evaluation of this prototype will follow shortly (chapter 5).

4.2 Snippet Factory

While investigating multidisciplinary meetings, we discovered that healthcare employees require a greater form of flexibility than workflows are designed for. To realize flexibility in a static environment such as workflows, we have several choices to consider. At what point exactly is flexibility needed? Research is conducted into making the static execution of workflows more flexible [1]. However we do not believe that flexibility on execution level is required, e.g. we do not expect that workflow instances have to be modified while they are running. We reckon that providing a means to create user-defined workflows will resolve most requests for flexibility. We therefore focused on providing a better way of creating workflows. This resulted in the idea of a second prototype, also known as Snippet Factory (or short SF). The Snippet Factory was intended to demonstrate how existing workflows could be reused, recycled or repurposed [21]. From [21] we know the reasons for sharing workflows (either in total or partial) are numerous and highly divers. As a result the sharing of workflows is becoming a more interesting topic every day. This section will discuss the ideas on this topic in more detail. Note that this prototype has never been implemented and therefore is strictly theoretical.

4.2.1 Goal

The goal of SF is to provide some sort of means for healthcare employees to quickly create workflows and load them into the workflow system (TTM). There are several options to consider here. We could provide the employees with the possibility of drawing their own workflow model (e.g. using the YAWL workflow editor). In general however, the drawing of workflows (especially those enriched with programming constructs) is a tedious job and can get quite complex. This problem could be resolved by creating a workflow editor which works with parts of other workflows. These parts are created by an expert which means the user is not bothered with programming details. The idea behind this (second) prototype is to provide the user with a possibility of "clicking together" a workflow without actual knowledge of the workflow management system behind it.

We want to obtain this kind of functionality by creating a system which can analyze previously created workflows and split them up in smaller parts. Besides this functionality we want to provide the system with a basic library of "common utilities" (exactly which functionality is not relevant at this moment). All of these separate pieces of workflows would become components, which can be reused to create a new workflow. The more ad-

vanced programming parts are then already created at the moment the library is initially built, removing this aspect of workflow creation for the users. In short, we want to create some means of taking existing workflows and recycling them to create a library of common components: the Snippet Factory.

4.2.2 Methodology

We want to create some means of disassembling existing workflow models into components, analyzing these components and storing them in a database where we can retrieve it without too much problem. The components, or snippets as we will call them from now on, can then be used to create new workflows. By allowing the user to create workflows with these snippets, the user is offered a great deal of flexibility at a relatively small cost of some added complexity for the programmers. The remainder of this section will describe the processes required to complete such a system.

We can obtain snippets in two different ways. The first method is to design our own snippets. Because workflow models can in fact consist of other workflow models, modelling of snippets can be realized by using any workflow model editor. A more interesting approach on obtaining snippets is to disassemble (parts of) other workflow models. The best method of doing so is to convert the usually plain XML data structure of a workflow model into a graph model. For this research we have chosen to try and convert the XML structure to a JGraph [16] model object. Although we will not discuss the model object in any more detail (assume it can perform the basic graph operations, like transformations and traversal), we will take a closer look at the conversion.

After the conversion of XML to JGraph‘s model object we can transform the graph as we wish. We can recombine snippets to form new workflows or remove certain parts of existing workflows. At this moment we also want to create a library to be able to save our current creations. The method for storing the snippets is irrelevant at this point and will not be discussed.

With this library of snippets and graph algorithms we are able to create a new workflow. The next logical step is to undo what we have done in the first step and convert the JGraph model object into its original XML format so that the workflow engine can understand it. In an ideal situation we can imagine that the workflow engine understands the model object itself (as workflow models are graphs), but for now this is not the case.

Because of time constraints we were unable to completely develop any of the previously described functions. The only step we managed to accomplish within the given time frame was the conversion of XML to graph model (and a bit of vice versa). The remainder of this section will try and explain how the system should have worked. We will do this by first taking a closer look at the XML structure YAWL uses to represent the workflow models. Finally we will see how the conversion can be realized.

On YAWL and JGraph The choice for JGraph is easy to explain. The original workflow modelling tool provided with the YAWL engine is in fact a graphical interface to JGraph.

Since the original authors of the YAWL system also use the graphical representation of the JGraph model within their own programming, it is expected that the workflow engine itself should be able also to handle these models. After some research into the source code of the workflow engine however, it seems that this is not the case.

The YAWL workflow engine has direct integration possibilities with its own provided modelling tool. The question remains if in the future this integration allows the uploading of the graph representation instead of the XML structure. We reckon that this is possible and should not be too hard to incorporate. The question remains however if such integration is desirable. Many arguments both in favour and against such integration can be given. We will not try and discuss all possible pros and cons at this time.

XML Format

The XML contains all details related to a YAWL-task (interestingly enough YAWL itself calls them "tasks") and are described in a nested structure per task. For the purpose of introduction we will shortly look at how such a task is described and what kind of information is present. For the sake of simplicity we have omitted some internal data which is stored as well, as it would only confuse the discussion. Note that the example below only depicts one process from the workflow model. The initial specification and namespace declarations have been omitted.

```
<task id="Start_Verwerking_4">
  <name>Start Verwerking</name>
  <flowsInto>
    <nextElementRef id="Selecteer_Client_7" />
  </flowsInto>
  <resourcing>
    <offer initiator="system">
      <distributionSet>
        <initialSet>
          <role>RO-8fb27836-c327-4efb-a362-28616b3a89a8</role>
        </initialSet>
      </distributionSet>
    </offer>
    <allocate initiator="user" />
    <start initiator="system" />
  </resourcing>
</task>
```

As shown we see that each task has a certain task id. This is a unique identifier within this particular model. Next it has some human readable name which is used for imagine generation and to display on the screen. The first interesting section then is called the *flowsInto* element. This element defines the edges between this vertex and other vertices in the graph. In this particular case there is only one (to *Selecteer_Client_7*).

The resourcing element specifies the offer, allocation and starting initiator. This basically means how this task is configured to be distributed. In this particular case, the (YAWL) system offers this task to the users who have a certain role (this was described in the *distributionSet*-element). Then a user is allowed to allocate a task to himself. The system then automatically starts the task at the moment the task is assigned to a certain user.

From XML to JGraph model and Vice Versa

The conversion of the plain XML file to the JGraph model object is a matter of parsing. For our prototype we have created a very simple algorithm to do this. The algorithm for converting the XML to the JGraph model first creates all vertices by parsing the XML file for all *task* elements. It stores the contents of each task in one larger variable for later parsing. After this first round of parsing, we now have x vertices corresponding to the amount of tasks within the XML file. The edges are then created by parsing the content of each vertex for its edges (namely the *flowsInto* element. After this second run we now have two simple lists, namely the vertices and the edges. These two combined form our graph. One of the default JGraph objects which is included in the distribution allows the creation of a simple directed graph based on these two lists. We have chosen to create a new class based on this directed graph implementation[5] class that can store the Task-object which we have created for our prototype. This way, we now have a JGraph DirectedGraph model representing our workflow.

The method of storing a JGraph model as XML data is slightly more difficult than the other way around. The topology of the graph is not the issue. Any traversal algorithm can create, per task, an XML structure representing the currently visiting task. Note that one should use an algorithm which only explores each node once. The difficulty with creating workflow models from snippets lies within the additional information. Once a snippet is created, certain information is stripped because it is too context-aware, for instance the role which can perform this task. We want to save snippets as abstract as possible. However, the YAWL engine does not support this level of abstractness within its models. For our prototype we have decided that the allocation and starting parameters (*allocate* and *start* in the XML structure) are static. The role section process should on the other hand be included in the design of a new workflow model.

Although we have not fully developed the previously mentioned algorithms, we assume that there should be no theoretical issues preventing such system to be developed.

Analyzing Snippets

Each snippet that can be stored within our library must be indexed with certain parameters. These parameters for instance can be the name of the snippet or the description (e.g. what does it do). Ideally, this analysis can be automated, e.g. the snippet can be sorted based on its ingoing and outgoing connections. However, because the workflow model within YAWL is not based on a scientific, but rather a business workflow, not all edges have attributes

[5]The official name for this class within the distribution was *DefaultDirectedGraph*

associated with them. This level of freedom is, as we have seen before, useful for modelling human actions, but it does also mean that sorting and indexing automatically becomes very difficult.

Our recommendation for this problem is that the creation of a snippet library should be conducted manually. In this particular case Topicus should create a basic set of building blocks to provide the healthcare organisation a basic set of instructions. The development of snippets separately is something that can be performed by a workflow modelling tool. This means that all individuals with a basic understanding of the workflow system should be able to create these snippets.

4.2.3 Issues and Resolutions

During the development of this prototype we ran into several problems which makes the implementation of this prototype quite complex. We have chosen to discuss only one of the major problems we found during initial research. Because we have not implemented this system further and spent rather little time on development we see no reason to include all minor problems here as they could be resolved during development by more experienced programmers. (There is no problem in theory, only in a practical sense.) The problem we want to discuss in more detail is that of the pre and post conditions of tasks.

Pre and Post Conditions The problem of the pre and post conditions connects directly to the assumption we made about tasks in the first prototype. Recall that we agreed that all tasks contain their own specific context and all attributes they require to be completed. We have seen also that at certain stages during the execution of a workflow model not all of that required information is present and we needed some method of transmitting information to tasks to complete their context. We created the *updateTask* command specifically for this task.

Assuming that a certain task needs attribute A, but we cannot determine A before another (second) task is completed. We cannot complete the context for this particular task until after the second task. In short, this means that (let's assume) the third task has a pre-condition of attribute A to be completed before it can be executed. We then need some other task, e.g. the second task to have a post condition which states that attribute A is now set and completed. Within regular workflow models this is not a problem and in most cases we can accomplish this by using the previously described update functionality.

Because we are now talking about snippets, possibly created by disassembling other workflows, these pre and post conditions now become a problem. Because we can combine tasks which potentially have a pre-condition about a certain attribute with other tasks that do not fulfil these pre-conditions, we can create workflow models which are unable to execute. Note that this a prototype problem and not a YAWL problem. (e.g. YAWL does not know these things about pre and post conditions.)

To resolve this, we have in general two choices. Either we can provide pre and post conditions to tasks in the library and allow users only to form workflow models which are

conditional correct. Or we need to eliminate the problem in general, e.g. only allow the creation of workflow snippets that do not have any pre-conditions. The first solution would be more ideal, allowing the users to be more explicit in their creation by allowing more complex snippets to exist. It however does introduce a new level of complexity during the creation of these snippets (e.g. users need to understand how they work and why some models won't work). In practice however, we reckon that snippets should not contain pre and post conditions to keep the modelling process as simple as possible. This does not mean that any workflows which are inside the snippet (remember that workflows can be constructed by combining workflows) should not contain pre and post conditions. Our recommendation is only that the snippet shouldn't have pre and post conditions (formally: all the pre conditions declared within the snippet should be met by post conditions within the same snippet).

4.2.4 Summary and Conclusion

Allowing users to create their own workflow models still has some unresolved issues. First of all, the XML structure used to store the workflow models should be transformed to some form that can be easily manipulated. We have chosen to use the JGraph model object form for this task. Next a better user interface should be created to allow basic transformations on the created graph representations. Once the workflow model is completed, it should be converted back into the XML format and uploaded to the workflow engine to be enacted.

The analysis of the newly created workflows on either snippet level or some higher level, is a difficult process to automate. For now we would advise to make the analysis of snippets a manual task, although in the future we expect that this could be automated. We think some form of pattern recognition on a large group of snippets could result in the creation of heuristics which could be used to determine the value of each individual newly created snippet. At this moment however, we do not have a large collection of snippets and thus would recommend manual analysis.

We have tried to develop a system which allows healthcare professionals to build their own workflow models. This allows them to assign certain case specific tasks to other employees or design their own healthcare treatment plans for clients. We reason that the creation of a whole complete workflow model was too much for a healthcare professional. The creation of a workflow model is difficult and complex and should in most cases be performed by a business expert. This prototype tries to simplify the process of creating a workflow model so that a healthcare professional, without having to be completely retrained, can create basic workflow models to suit his needs. To accomplish this, we think that the use of a snippet library is the best practice. Within this library basic components should be created which can be combined to form one workflow model.

Chapter 5

Prototype Evaluation

The initial intentions were to hold user evaluation session. However since the prototype resulted in being rather abstract and more to be a proof of concept, we decided to evaluate our prototype with the developers, designers and analysts of Topicus. Initially, two separate evaluation sessions were arranged: The first was held with two analysts and we focused on the use of the system in the healthcare environment. The second session was held with developers and focused at the technical potential. Unfortunately, this session was held with only three persons, of which two participants were biased against workflows. Because at this moment we only had five opinions, we decided to organize a third session, inviting the whole business unit. The remainder of this chapter describes the results of these three separate evaluation sessions in more detail. First we will take a look at the goals and methodology we used for our evaluation and look at consistency of the participants group.

Goals and Methodology For each of the sessions a plan was devised and a list of questions which should be answered was derived. Since our prototype was more of a proof of concept we felt no need to create statistical evaluation questionnaires. For our evaluation we have used an adapted version of the *teach back* methodology as described in [33]. Teaching back basically means that we introduce the concept version of our prototype system and then ask the participants to explain what functions they expect. This differs from the described method on basis that we have only one setting, where the original teach back method uses two (one concept and one actual system is evaluated). We adopt the method of teaching back and use it by asking the participants how they would improve our suggested system. By asking certain direct question we hope to get a discussion started. To accomplish this we created a few open questions which were asked after the initial presentation. The initial presentation was used to illustrate the prototype and its functions (similar to the description in the previous chapter). Although the method of conducting the evaluation differs slightly from each other, we have tried steering the discussion in such a way that at least the following questions were answered:

- Do the participants understand what the prototype does?
- How should workflows be used in the healthcare environment?

- Were any opportunities missed for workflow usage in this particular setting?
- Do the participants like the idea behind the prototype?

Each of the following sections will discuss most if not all of the above mentioned questions. Note that these questions were used to start up a discussion about the prototype, so not all of these questions are answered in all sections. For us, the most important goal of this prototype is to make the participants aware of the workflow uses and try to discuss other potential applications we have not thought of during this research.

Participants For our evaluation, we have used employees of the Topicus company. Within Topicus we have basically three different types of experts working on each project at any given moment. These are analysts, designers and developers. Before we will discuss the outcome of the evaluation we first will introduce each function individually.

Analysts are generally not concerned with programming details of an application, but are responsible for the completeness of an application with regards to its functionality. Analysts create functional designs, which are then used by developers to build the actual system. In these functional designs all the functions are described and all actions that users can perform are covered. During the development of a system it is the role of an analyst to "be" the end-user of a system. Within Topicus Care, some of the analysts are also responsible for testing and may be part of the design team.

Designers are responsible for the overall appearance of an application. They ensure that all pages appear to form one system and that the interfaces are well understood. Every now and then they are assisted by analysts and designers to redesign a certain piece of an application.

Developers are the builders of an application. Based on the functional design created by analysts and in accordance with the design team, web-pages are created for the purpose of introducing certain functionality. They are also responsible for fixing any problems within the applications (e.g. bugs) and maintaining a certain standard for their code.

5.1 Evaluation by analysts

The first evaluation session was held with two analysts of Topicus Care. This session started with a general introduction into the research, followed by an explanation on basic workflow terminology. Then a demonstration was given of the prototype and used to open the discussion about the prototype. Both analysts are business professionals which resulted that most of the discussion involves the business potential (of workflows). The remainder of this section describes the results of this evaluation.

During this evaluation we have discussed some very specific business related questions. Among these questions were the efficiency gain of workflows versus case management situations and the adaptation of such a system within legacy-based environments. Although we reckon that these are very valid questions and should be researched in more detail prior to

the development of such a system, we find that they are not as relevant for this particular research. We have therefore chosen not to include these questions in the total results.

Task Progress The analysts reckon that it would be highly desirable to have some graphical overview of total task progress in the users interface. This means that it would be nice to actually see the overall picture of what tasks are completed and what tasks are still in progress. Figure 5.1 provides an example on what the analysts meant during this discussion.

Figure 5.1: A possible graphical representation of task progress. The red colour represents completed, green stands for current and white is for future tasks.

Task Progress Monitoring Noted was the fact that this kind of workflow application could resolve the problem which larger organisations begin to experience. In healthcare organisations with more than 10,000 clients, tracking the progress of each individual client becomes challenging. With the application of the workflow system in the way the prototype demonstrates this problem is nearly completely resolved. There is one central point which keeps track of all progress for all clients.

Statistics Business workflow systems usually generate a decent amount of statistics. The YAWL workflow engine keeps track of a lot of separate statistics, like the amount of time a task requires for completion. Based on these statistics business procedures can be adjusted and fine-tuned to enhance productivity. It was suggested that these statistics, which were quite underdeveloped for this prototype, should be brought into view. If not within this user-oriented client, then some other administrative client. Healthcare organisations are increasing in size and this kind of performance measurement is highly in demand.

Authorization Because healthcare organisations work with a lot of data that fall under privacy law and legislation, they are bound by all kinds of rules and regulations to prevent unauthorized access. Workflow based systems could be used to enforce these rules as well. One of the possible methods how workflows could accomplish this is by restricting access to data for a certain user, until the workflow system assigns a task which needs the user to access that specific data. The workflow engine (or some other system) would then temporarily grant the user access to this data, allowing him to perform his task. At the moment the task is completed, the access is revoked leaving the system secure. More research should be conducted on how to actually realize such controlling systems.

Other Remarks There were some questions asked with regards to the application of such a prototype with regards to care-providing tasks. We have tried covering these tasks within our prototype, but the overwhelming amount of different standards and methods used by the various multidisciplinary users made it impossible to create a system which would suite all. Besides this variation in methodology, care-providing tasks have the problem that they cannot be classified as "100 % completed" or any other numerical representation for the amount of completion. This means that we have a strict system, which only accepts the facts that a task is complete or not, that now must deal with these variations. We reckon that some assistance can be provided, although this particular topic needs more research.

It was also noted that, in addition of the task progress monitoring described earlier, it would be nice to have some of history options with regards to the task. Based on the history of task completion, users can, if necessary, contact the user that completed an earlier task for help or comments.

On a final note, it was remarked that the method of listing tasks in the interface was not optimal. Instead of showing and sorting on task names, which is done now, it would be better to list tasks on basis of the client's context. For instance, use the clients name for sorting and displaying.

5.2 Evaluation by developers

The second session was organized to be a Topicus-wide evaluation session of the workflow application. We sent out invitations to ten different developers (two per business unit, not all business units were included) and asked them if they would come over and discuss the workflow application. No responses per email arrived and at the session self, only three developers showed up. Two of these developers originated from the same business unit and shared similar thoughts with regards to workflow use. The remainder of this section describes the opinions of these developers.

Multi-user task tracking One of the favourable aspects of using workflows is that tasks are now separate entities within the system. Tasks can be monitored and treated as individual components within the system. This allows a task to be traced back several users and should improve overall efficiency of "task-based" application usage.

Increased Complexity Where analysts saw a lot of potential. Developers mostly saw problems. They reckon that use of workflow does provide a lot of freedom in modelling and acting. But this comes at the cost of increased complexity. The idea behind workflows of providing a model layer to provide a suitable method of describing the business logic was appreciated, although the amount of complexity put the developers off. In short, the developers are slightly worried about the additional complexity with regards the usage of workflows.

Unused Functionality The developers motivate their concerns about additional complexity to allow freedom of modelling with the statement that the introduced freedom will not be used. In short, the introduction of the workflow modelling layer provides loads of freedom for the users at an increased level of complexity for the developers. Although the developers agree with the concept, they claim that this modelling will not happen in their current environment. Most likely a workflow model will be created once, or twice, and then always used over and over again. The developers reckon that in such situations it would be best to keep the code as well as the usage of an application as simple as possible and accept the fact that this particular level of freedom is unwanted.

In conclusion, although the developers agree with the concept of workflow technology as a method of supporting users and treating tasks as individual pieces of the complete system, they advise against using workflows. Because the overall complexity of a system increases and the gains are very little or possible the freedom of modelling isn't used at all, the developers advise to keep the system as simple as possible.

5.3 Group Evaluation at Topicus Care

The last evaluation that was held was with employees of the Topicus Care business unit. Because of the expected number of people was over 15 (eventually 16 people showed up) we decided it was best to create a questionnaire so that all results could be gathered on paper. We also grouped each person on their role the organisation. For this evaluation we had developers (8), analysts (6) and designers (2). The questionnaire was made up from five different questions. The common goal which all of the questions shared was to get ideas rolling within the unit and to think ahead. Prior to the evaluation an overview was given of the prototype, focusing on the functionality it provides and how the workflow engine was used. For this section we will look at each of the five questions that were asked and the results that were found. These questions were:

- What is your first impression of the prototype?
- Do you expect that the concept demonstrated in the prototype could be applied within applications current developed by Topicus?
- Do you reckon that we missed any functions or possibilities in this prototype? And if so which ones?
- What kind of additional functionality do you expect by introducing workflows?
- Do you have any other remarks with relation to the prototype or the use of workflows?

5.3.1 Analysts

First Impression The idea of using tasks in the sense that is shown by the prototype matches that of the modern healthcare organisation. Although the prototype is lacking

certain client specific information it demonstrates how these tasks could be represented. The system is flexible enough to cover every small task which is being performed within the healthcare environment. Additionally an increasing number of healthcare organisations are creating process models which could be easily mapped to the prototype.

The interface could use some additional work, searching for instance would be highly favoured in larger settings. Overall the prototype was found clear on its function and how it behaves. Some questions were raised with regards to the appearance and most importantly, the method on how tasks are displayed. The first comments stated that the task name might not be the best criteria to display a task, but rather the clients name or some other part of the client's context.

Usage Expectation On this question a lot of different answers and opinions were returned. Most of the analysts agree that the concept demonstrated within the prototype is useful and should be incorporated within the systems that Topicus develops. The difference in opinions originates in the discussion on how and when the concept should be integrated. First of all some state that current applications could use the concept of using workflows as task guiding systems as long as the integration of the two systems are seamless, e.g. it should appear to be one. Others say that the introduction of such a system in this phase is unwise and that future systems should be designed with such workflow techniques as a basis. In other words, some say that the introduction of workflow support could be used right now, while others claim that it would be best to wait and create a new "series" of applications based on this particular workflow technology.

Alternative Possibilities The concept of creating tasks which involve different users, aids the concept of task transferability. Task transferability means that one task is partially completed by some user and then passed on to the next which then completes it. Although the prototype already demonstrates how this could be accomplished, it was reckoned that the system should be a bit more responsive to such transfers. It was deemed a good idea to have the system notify the previous user of the fact that the task was successfully transferred to the next user.

Other possibilities included the support of multidisciplinary teams, which we have discussed earlier in this thesis, and the automatic creation of workflow models on basis of use-cases. Use-cases are created by analysts to completely model the functionality of a system, thus it would make sense to create a use-case to workflow model conversion system. Although we have not fully investigated this idea, we reckon that the snippet factory described earlier could be a valuable asset to this problem. The snippet factory contains a lot of "small" tasks and pieces which could be directly mapped to a use-case.

Additional Workflow Functionality The one thing that was completely lacking in our prototype was the use of statistics for management purposes. The workflow engine contains certain features, like average execution time, which are invaluable to managers. With these statistics they can monitor task progress and act in case a task is running out of time or

takes too long. On a higher level, they can determine which tasks are too complex and take up too much time.

Another item that was lacking in the prototype is the detection of task progress. At this moment a user first manually needs to assign a task to himself before the task progress is monitored. In future systems it would be highly desirable that this step became obsolete, meaning that the user can just start with a task and the system notices that this particular task is being started. For the sake of simplicity we have opted not to try and accomplish this in our prototype, but we reckon that such an addition would be highly favourable for the overall system performance.

Finally, it was noted that the task manager itself should display more details about tasks. It should contain the amount of tasks that are currently open (in a more "intelligent" manner) and display which tasks are currently being executed by colleagues. These small details make the planning on who should accomplish what task slightly easier and it provides the actual users with a reference to their colleagues in case they need assistance.

Other Remarks Some analysts mentioned that the freedom of workflow modelling would be highly suitable for this particular environment. Although there was no real argumentation present on the questionnaire, we reckon that these comments favour our suggestion for the snippet factory. Note that during this evaluation session we have not discussed the snippet factory or the manipulation of workflow models in any way. Therefore we can conclude that this particular topic peaks the interest of (at least) some of the developers.

Another issue that was raised (also in section 5.1) is the point of authorization. It would be nice that during the execution of a task, non relevant information is shielded from the user. This increases the level of privacy and security which are very important for healthcare organisations.

5.3.2 Designers

First Impression The idea behind the prototype was well received and considered useful. The amount of tasks and detail of each individual task is something that should be given more attention during development. On a design note, the interface was decently designed although the different type of buttons should be changed. The simplicity of the interface is a bonus.

Usage Expectation The concept demonstrated by the prototype could be used within Topicus applications. However, the problem is that it might only work with pre-defined tasks and this lack of flexibility might become a problem in the future.

Alternative Possibilities The prototype misses a lot of managerial functions, like assigning tasks to a person and making priorities. Also logging of completed tasks should be done for each user to see what user completed the task prior to the task of the current user. This logging will ensure that any problems during task execution can be resolved quickly.

Additional Workflow Functionality The previously mentioned history of a task-path (*Other Remarks* in section 5.1) is something that should be implemented. Also when switching from context from one task to another, it should be made clear in which client's file (context) the user is currently working. At this moment there is no visual indication that the swap even succeeded. The menu-interface also should contain some statistics for the user self, like how many tasks are left and which tasks require immediate attention (priority).

Other Remarks The overall design of the task manager is sufficient, but should be better integrated in the related system. At this moment it appears that the task manager is a completely separate system, although without the underlying system it would have no function. It was suggested that the task manager should appear in a pop-up window and should allow a bit of navigation with regards to the tasks. This navigation could be combined with task history or additional task information. In short, the task manager should contain more context specific information and should offer this to the user in a more visually attractive manner.

5.3.3 Developers

First Impression The issue that was most dominant on the evaluation forms was the fact that the prototype and the related system were too separated. Although some argue that this is a good thing (*Because it is separate, you are not forced to use it.*) most find it distracting and opt for a more integrated look. The simplicity of the prototype did receive good response. It was stated that task-driven assignments are preferred to data-driven assignments as most of Topicus systems adopt today. The prototype adds a global overview which gives the user a better understanding on why he is performing the job he does. The scenario that was chosen for the demonstration however, did little justice to the overall system.

The fact that the code becomes more open and simplistic is also a bonus, although most developers expect that the way it was presented might not have been the best method.

Usage Expectation Most developers that were interviewed had some form of previous experience with workflows. Most of these experiences however did not leave any good impressions and the overall enthusiasm prior to this session was rather low. The results of the evaluation were more positive, although a certain level of hesitation must be noted. We reckon that the developers are worried about the additional implementation issues and the introduction of increased complexity.

The developers agree with the analysts on the fact that using workflow systems is a good method of dealing with a large number of clients.

Alternative Possibilities In a completely different setting, it might be the case that tasks are not selected by employees themselves but rather assigned to them by some supervisor. In that case it would be a welcomed addition to have an overview of the tasks that should be completed on a particular day or in a particular week. Some developers think that

besides these display issues it would be nice to have some form of planning structure or an additional method of reasoning with tasks. For instance, it could be desirable to plan certain tasks from a complete chain ahead so that the total execution time is as limited as possible. Such situations could benefit from introducing priority tasks and the use of other managerial and statistical functions.

Additional Workflow Functionality Besides a fairly large amount on functional advise on how the interface should function a few interesting results came from this question. The developers expect that a workflow system like this could only function if a certain minimum standard is obtained. Tasks should be allowed to be prioritized or "made urgent". Workflow models should be allowed to be complex [1]. Also there should be more logic involved in creating the workflow models, like conditionals based on the underlying databases.

Also, the prototype should verify the completion of a task to a certain standard. During the development of our prototype we intentionally did not focus on this, because creating heuristics for verifying the completion of a task is rather difficult to fully implement. (Providing of a clients name does not mean it is the correct name.)

Finally, some developers wanted the possibility to create ad-hoc workflows. During this evaluation we did not mention the snippet factory, therefore we reckon that the creation of ad-hoc workflows is something that should be researched further.

Other Remarks Some of the remarks were about the integration of the prototype. Some suggest that the integration should be created on a deeper level (e.g. make it part of the underlying system), others approve of the clear separation. Overall the concept was received well.

Additional remarks were made with regards to the requirements of a workflow system. It was reckoned that the system should complete the shortcomings of the underlying system in such a way that even without the task-based thoughts it still would improve the overall system. Although this is a noble goal, we reckon that the best strategy of using workflow technology is to keep the overall complexity as simple as possible. It was suggested that the workflow system could be used to store partially completed forms for further completion. We understand why this would be desirable, but doubt if this is the responsibility of the workflow system. We reckon this "problem" of saving partially completed forms is more the responsibility to the web-application it is connected to.

5.4 Conclusion

The analysts in the first section of this chapter value the possibilities which workflow technology created. Workflows can cope with a large number of clients and yet can provide individual progress reports. The possibility of generating statistics about these tasks is also a highly valued item on their lists.

[1] For our prototype, we kept the complexity of the workflow model to a bare minimum.

The first group of developers is slightly more sceptical with regards the workflow use. They can see how it would benefit the user, but doubt the user will fully utilize all the possibilities which workflow technology introduce. Developers are slightly more conservative and have stated that the best approach forward is not using workflows.

For the first two sections we must conclude that there is standoff between developers and analysts. Analysts claim that the usage of workflow systems possesses a certain potential which should not be ignored. They allow for organisations to monitor and improve performance of tasks and keep track of vast amounts of clients simultaneously. Developers also recognize the potential, but doubt that the complexity workflow communication introduces outweighs the benefits that can be obtained. They imply that the current applications should be kept as simple as possible to provide the best support for what the users want.

The overall Topicus Care evaluation also gave some interesting results. Here as well we see a slight stand-off between the developers and the analysts, although both groups provided some very interesting points.

The analysts see a lot of potential within existing as well as future systems. They do recognize however the problem that the increased complexity might become an issue in the future. They see several possibilities for workflow technology, ranging from training sessions to the optimization of business processes using the statistics generated from the workflow system. Analysts expect that the creation of ad-hoc workflows will also become a point of interest.

The developers are slightly more concerned with the increasing amount of complexity imposed by introducing workflows. They acknowledge that the method of task-based applications instead of data-driven applications is the way forward, but are not sure if workflows are the best method of accomplishing this.

The designers agree with the concept of workflow application as demonstrated within the prototype. They reckon that additional information about each task, like which task in line it is, which tasks have been completed and which are yet to come, is vital to the success of such a system. The graphical representation of progress will help to better understand why certain tasks need to be performed.

Overall the system was accepted by all with some remarks. We therefore can conclude that the application of workflow technology as demonstrated by the prototype is useful, although there is still room for improvement. We must admit however that our usage of workflow technology is only one of several possibilities and thus it is possible that other solutions might be better suited for the healthcare environment. For this experiment however, we must conclude that the concept of workflow use is clear and well understood.

Chapter 6

Conclusion

At the end of our research we first of all can conclude that the potential of workflow usage within the healthcare domain is vast. We have chosen to limit ourselves to the healthcare domain found in residential care homes. We performed an analysis of the processes that could be found within this domain. A distinction can be made between care-providing tasks and administrative tasks [1]. Most of the health-providing tasks are performed by healthcare professionals and nearly all have their own method of performing these tasks. Making one model to cover all possible situations would be impossible. Therefore we have chosen to focus our attention on the administrative tasks in this domain. The field of administrative tasks is also large, ranging from simple tasks which can be automated with ease, to rather difficult tasks where human interaction is required. Sometimes it is a matter of changing the client's personal information like an address and sometimes it involves determining the level of healthcare a client requires. Although most of the tasks found within this administrative field should be performed by a healthcare professional, we found that some are highly suited for automation. This answers the second research question, which focused on the type of processes within the healthcare domain that could be supported by workflow systems (RQ2).

Before we could try and determine how to support the tasks, we have investigated the domain of workflows themselves. There are a lot of different workflow systems available on the market today. By analyzing what a general workflow system does and how it functions we have concluded that workflows are best suited for rather static environments where processes do not change frequently. Although workflow models themselves can be altered quite easily, an executing workflow instance is very hard to change after it has started. We therefore should use workflows in environments where processes do not change very often and suddenly. Workflow systems are capable of handling large amounts of clients at the same time. They keep all clients separated by creating individual instances for each client. The exploration of the workflow system as a whole, as well as determining the strengths and weaknesses of workflows are part of the answer of our first research question (which is meant to determine how workflows work and what their strengths and weaknesses are (RQ1)).

We learned from our workflow analysis that there are several functions a workflow system

[1] We have chosen to use the term task as an equivalent to process.

can perform. One of these functions, and the one we chose to create our prototype upon, is that of a guidance tool designed to help out on the interface level of an information system. An example of the current unaddressed issues (subsection 2.2.5) found within our healthcare domain was the lack of overview. Therefore we have chosen to use a workflow system as monitor and guide for the administrative tasks by storing client information and task progress details in our workflow instance. This is the first part of the answers to the fourth research question (RQ4). As a proof of concept, we created a working prototype to demonstrate the found functionality in combination with one of the products Topicus Care [2] has developed called TisWeb. TisWeb, a web-based application, is used to provide easy access to client information. TisWeb was designed to be an intuitive interface to help healthcare employees in their daily routines. We have used this interface to develop our prototype and evaluated the new combined system after. During this evaluation we have found great potential in the field of task tracking and progress monitoring as well as generating statistics with regards to these tasks. Overall the evaluation showed that our experiment with the created prototype was successful with regards to introducing workflow technology into the healthcare environment.

Besides trying to determine the workflow potential for the above mentioned tasks, we were also asked to perform an in-depth analysis of the multidisciplinary meeting and try to develop a method of supporting this particular process. By requesting information from various agencies and even interviewing a manager who frequently attends such meetings we have discovered that multidisciplinary meetings within the residential care home domain are rather important in determining the healthcare requirements for a certain client. A multidisciplinary meeting, where some of the healthcare professionals as well as relatives from a client attend, is more of a social process with the common goal to keep the "quality of life" to a certain standard. In relation to our research we must conclude that multidisciplinary meetings are used for information gathering from a clients representatives and thus require very little process support. Based on the information that was gathered we did however answer the third research question (RQ3).

The only additional requirement we could determine on basis of what we have learned is that the multidisciplinary meeting can be a source of ad-hoc additional tasks for healthcare employees. Healthcare professionals would welcome a possibility for the ad-hoc creation of small "treatment-plans" (or short-term ending procedures) and keep a track of their progression. We have translated this idea into a system which allows a quick generation of workflow models based on already designed components, called snippets, to allow a workflow model to be "dragged together" without too much effort. This overall system is called the Snippet Factory and is, in sorts, our answer to where workflow systems can be used to support the multidisciplinary meeting. This is the second part of the answer to the fourth and final research question (RQ4).

We can conclude that there is a lot of potential to be found for workflow systems within the healthcare domain, but careful considerations on where and how to use them is very important. The healthcare domain found in residential care homes is rather unique and therefore hard to compare to any other type of healthcare (with the notable exception of

[2]Topicus Care focuses on developing applications for the domain of residential care homes.

home care). This means that studies performed on a different part of the healthcare domain do not always provide good additional information.

During our research, we have made certain limitations to our research and choices which are not well supported. The reason why we had to choose at certain moments is that there was not enough time to completely explore all possibilities. As a result, certain choices were made to the best of our knowledge but in most situations we are confidant that we eventually made the right choices to provide a solid basis for future research. The last chapter of this research focuses on the limitations and choices and recommends where future research should be conducted.

Chapter 7

Future Work

During the research for this thesis, we have encountered several issues that could not be resolved in our given timeframe. Because of this certain assumptions had to be made (e.g. during the prototype creation) in order to proceed with our research. For this research we have tried resolving the issues to the best of our knowledge and overall we can state that it went rather well. For the remainder of this chapter we will look at some of the found issues during our research.

Care-providing Tasks Originally some questions were made about the application of workflow systems on care-providing tasks. Although later it was decided to leave that topic for a later research there are certain aspects we would like to point out. First of all, the modelling of care-providing tasks is already done in hospital environments in specific disciplines like oncology. These models are highly useful and could be used within workflow systems as they are now. The problem with the residential care home domain is that there are too many changes to the well-being of a client on a day to day basis to be fully modelled within just one workflow model. Prior to this research some ideas were put forward about dedicating a complete research project to the modelling problems we could encounter within the domain of care-providing tasks. To complete this research we chose to look into workflow support for administrative tasks instead. The modelling of care-providing tasks is something that could be highly useful, but a high level of difficulty should be expected since workflows themselves are barely meant to be "flexible".

Large Scale Implementation During the prototype implementation we have designed our system to handle only a small number of clients at a time. Our proposed system does not use any form of sorting and grouping in clients to keep the system itself organized. We suspect that a large amount of clients introduced into the system would make the interface unpleasant to work with, as the list of clients would be too large to handle. Although the concept of the system should not have any difficulties dealing with larger numbers, our interface was designed to handle ten to twenty tasks at most. To keep such a system in working order, we suggest filters or grouping mechanisms should be created. However, just the random application of such layers would be unwise, as there are no heuristics on

when someone should be filtered and when not. An exact approach on how to resolve these problems should be researched separately along with the problem how to treat a large number of clients simultaneously.

This may not seem like a problem now as most healthcare organisations are creating smaller units at the moment, but we expect that a more global perspective of the healthcare provided to clients would be welcome. We acknowledge that maybe the healthcare organisation itself might not be interested in this overview, but e.g. insurance companies and the government might. We recommend future research on this particular topic to determine the potential of providing large scale systems.

Pattern Recognition in Snippets As described in section 4.2 we have tried to develop a plan for automatic workflow model analysis. The major problem with this particular field however is that we do not have any baseline or dictionary on which we can evaluate the found model snippets. To be able to automate this function, research should be conducted in the characteristics of workflow models found within this domain. Based on this research heuristics might be derived which can be used within automatic model analysis. We think that a large library of workflow models should be created and annotated. With annotation we mean the manual selection of models (or parts of models) that are considered "useful" and suited to be reused. Determination of whether a certain model (or part of a model) is deemed "useful" or not is something that must be researched as well. If this annotation is complete however, it should be possible to discover patterns throughout the useful sections of each workflow model. If these patters were to be trained to a pattern recognizer (Hidden Markov Model, or any other) it should be possible to create a workflow analyzer capable of marking interesting sections of workflows. The method on how to accomplish this however, should be topic of future research. For more information about pattern recognition or HMM's, see [22].

Summary and Conclusion

This research has touched a lot of different topics. Sometimes we had to make certain decisions without complete knowledge about the situation. This section described some of the aspects we have encountered during this research and provides some suggestions on where future research should focus on. Without any doubt some serious issues have been forgotten and it might also be that some issues were deemed irrelevant while they actually turn out to be important. For now we can only provide recommendations for the future and with that statement we would like to end this chapter, and our research.

Acknowledgements

This research lasted nearly a year and during that time a lot of people have helped out to complete this project. In the next few lines I would like to thank these people for their support during this research.

First of all my graduation committee from the University of Twente; Betsy van Dijk for her weekly updates and feedback during the actual writing of this report. Dirk Heylen for his feedback at the report as a whole.

I want to thank the Topicus Company for giving me the opportunity to do this project. I had a wonderful time doing most work for this research there. Some Topicus (and former Topicus) employees need mentioning:

- Harald Dannenberg, as part of my graduation committee, for providing feedback and comments on my ideas while they were forming.
- Pepijn Noltes for being invaluable - almost every day - and providing a solid starting point with my prototype.
- Leo Eijkelenkamp and Hajo van Ravenswaaij (analysts of Topicus Care) for allowing me to discuss ideas at a very early stage.
- Sander Hofstee and Wesley Oosting. Without them the prototype would not have worked and would have looked horrible.

From the *Atlant Zorggroep* I would like to thank André Middelkoop for his support during the investigation into the healthcare domain and providing me with many pages of documentation regarding all aspects of the multidisciplinary meeting. Without this information I still would be wondering how certain processes actually are performed.

My father, Anjo Pothoven, for reviewing this entire document in nearly no time and providing me with page after page of red comments.

Last but not least I would like to thank all others I have forgotten to mention. Some have helped me writing the report, others have pushed me to (finally) finish it: you know who you are, thanks a lot!

Bibliography

[1] Michael Adams, A.H.M. Ter Hofstede, David Edmond, and W.M.P. Van Der Aalst. Worklets: A service-oriented implementation of dynamic flexibility in workflows. In *Coopis*, pages 291–308, 2006.

[2] Jrg Becker, Marc Gille, Michael Zur Muehlen, and Dr. Marc Gille. Workflow application architectures: Classification and characteristics of workflow-based information systems. In *in: Fischer, L. (Ed.): Workflow Handbook 2002, Future Strategies, Lighthouse Point, FL*, pages 39–50, 2002.

[3] Lindsay Bradfort and Marlon Dumas. Getting started with yawl, *http* : *//yawlfoundation.org/yawldocs/gettingstartedwithyawl.pdf*, visited June 11 2009.

[4] Topicus BV. The topicus bv website, *www.topicus.nl*, visited november 11 2009.

[5] A. Dwivedi, R.K. Bali, A.E. James, and R.N.G. Naguib. Workflow management systems: the healthcare technology of the future? volume 4, pages 3887–3890 vol.4, 2001.

[6] J. Emanuele and L. Koetter. Workflow opportunities and challenges in healthcare. Technical report, Siemens Medical Solutions, USA, 2006.

[7] *The ApacheFoundation*. Apache wicket, *http* : *//wicket.apache.org/*, visited december 2 2009.

[8] *The YAWLFoundation*. Yawl workflow system, *http* : *//www.yawlfoundation.org/*, visited november 16 2009.

[9] R. Haux, C. Seggewies, W. Baldauf-Sobez, P. Kullmann, H. Reichert, L. Luedecke, and H. Seibold. Soarian(*tm*) - workflow management applied for health care. *Methods of Information in Medicine*, 42:25–36, 2003.

[10] *IntermountainHealthCare*. Intermountain healthcare website, *http* : *//intermountainhealthcare.org*, visited february 9 2010.

[11] David Hollingsworth. The workflow reference model, Jan 1995.

[12] Vojtech Huser. *Using Workflow Technology for Modeling of Clinical Processes. (Chapter 5: RetroGuide Evaluation)*. PhD thesis, University of Utah, 2008.

[13] *RedHatInc.*. The hibernate project, *https* : *//www.hibernate.org/*, visited december 2 2009.

[14] Kepler. The kepler project, *https* : *//kepler − project.org/*, visited november 11 2009.

[15] F. Leymann and D. Roller. Workflow-based applications. *IBM Syst. J.*, 36(1):102–123, 1997.

[16] *JGraphLtd*. Jgraph - visualize everything, *http* : *//www.jgraph.com*, visited january 6 2010.

[17] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.

[18] Timothy McPhillips, Shawn Bowers, Daniel Zinn, and Bertram Ludaescher. Scientific workflow design for mere mortals. *Future Generation Computer Systems*, 25(5):541–551, May 2009.

[19] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[20] David Pinelle and Carl Gutwin. Supporting collaboration in multidisciplinary home care teams. In *Proceedings of American Medical Informatics Association (AMIA) Annual Symposium 2002*, pages 617–621. Press, 2002.

[21] Tristan Pothoven. Improving workflow searches in bioinformatics using different perspectives. *Human Media Interaction Website, http* : *//hmi.ewi.utwente.nl/verslagen/capita − selecta/CS − Pothoven − Tristan.pdf*, 2009.

[22] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[23] Xiping Song, Beatrice Hwong, Gilberto Matos, and Arnold Rudorfer. Understanding and classifying requirements for computer-aided healthcare workflows. *Computer Software and Applications Conference, Annual International*, 1:137–144, 2007.

[24] SpringSource. The spring framework, *http* : *//www.springsource.org/*, visited december 12009.

[25] James Stewart. *Calculus: Early Transcendentals 5th Edition*. Thomson, 2003.

[26] *The MyGridTeam*. The mygrid project, *http* : *//www.mygrid.org.uk/*, visited november 11 2009.

[27] S. Van Den Hoogen. Een methode ter vergroting van het it implementatiesucces in zorgorganisaties. Master's thesis, University of Twente, Faculty of EEMCS, Department of Electrical Engineering Biomedical signals and systems., 2007.

[28] W.M.P. Van Der Aalst. The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[29] W.M.P Van Der Aalst, L. Aldred, M. Dumas, and A. H. M. Ter Hofstede. Design and implementation of the yawl system. In *In: Proc. of CAiSE*, 2004.

[30] W.M.P. Van Der Aalst, A.H.M. Ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003.

[31] *Centraal Bureauvoor deStatistiek*. Centraal bureau voor de statistiek, *www.cbs.nl*, visited july 27 2009.

[32] $A + OVVT$. Evv profiel 2009, $http://www.evv - opleiding.nl/$, visited november 16 2009.

[33] I. Wassink, P.E. van der Vet, G.C. van der Veer, M. Roos, and E.M.A.G. van Dijk. New interactions with workflow systems. In L. Norros, H. Koskinen, L. Salo, and P. Savioja, editors, *Designing beyond the Product-Understanding Activity and User Experience in Ubiquitous Environments*, volume 258 of *VTT Symposium*, pages 349–352, Helsinki, Finland, October 2009. VTT.

Index

Appendix A

Walkthrough Figures

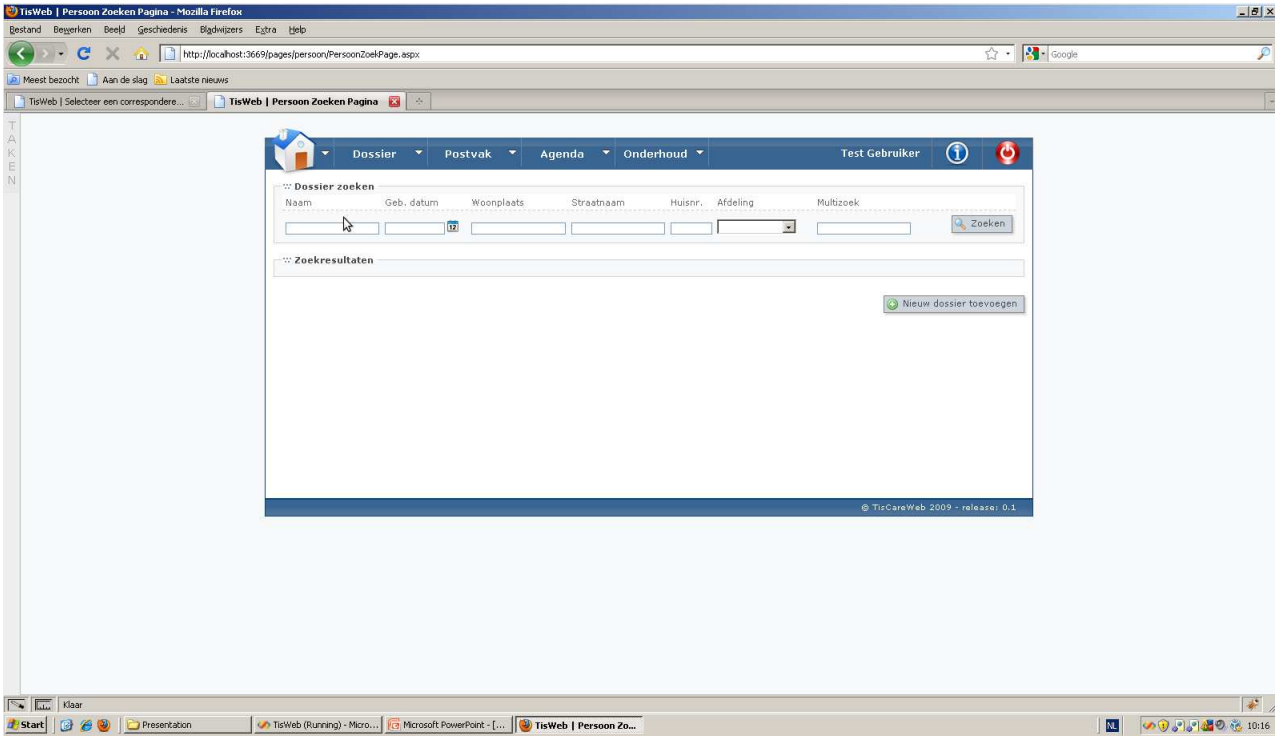

Figure A.1: Main screen with closed Task Manager

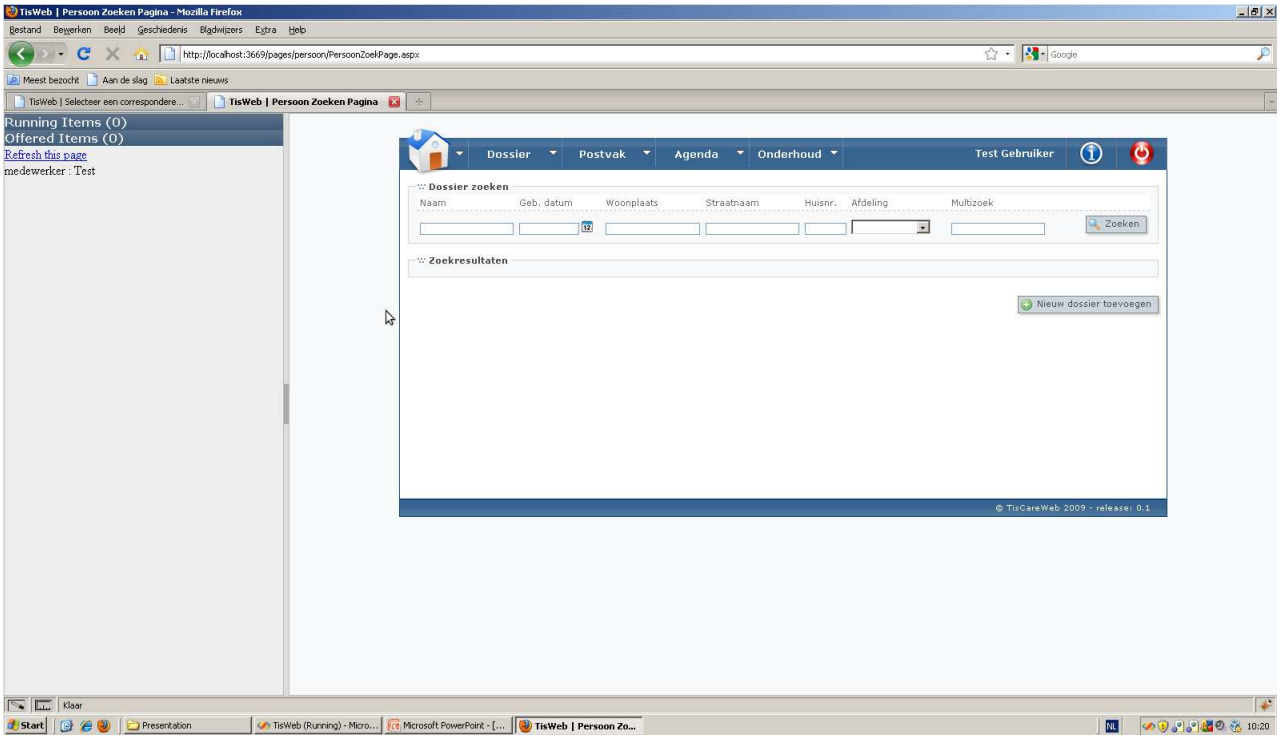

Figure A.2: Main screen with opened Task Manager

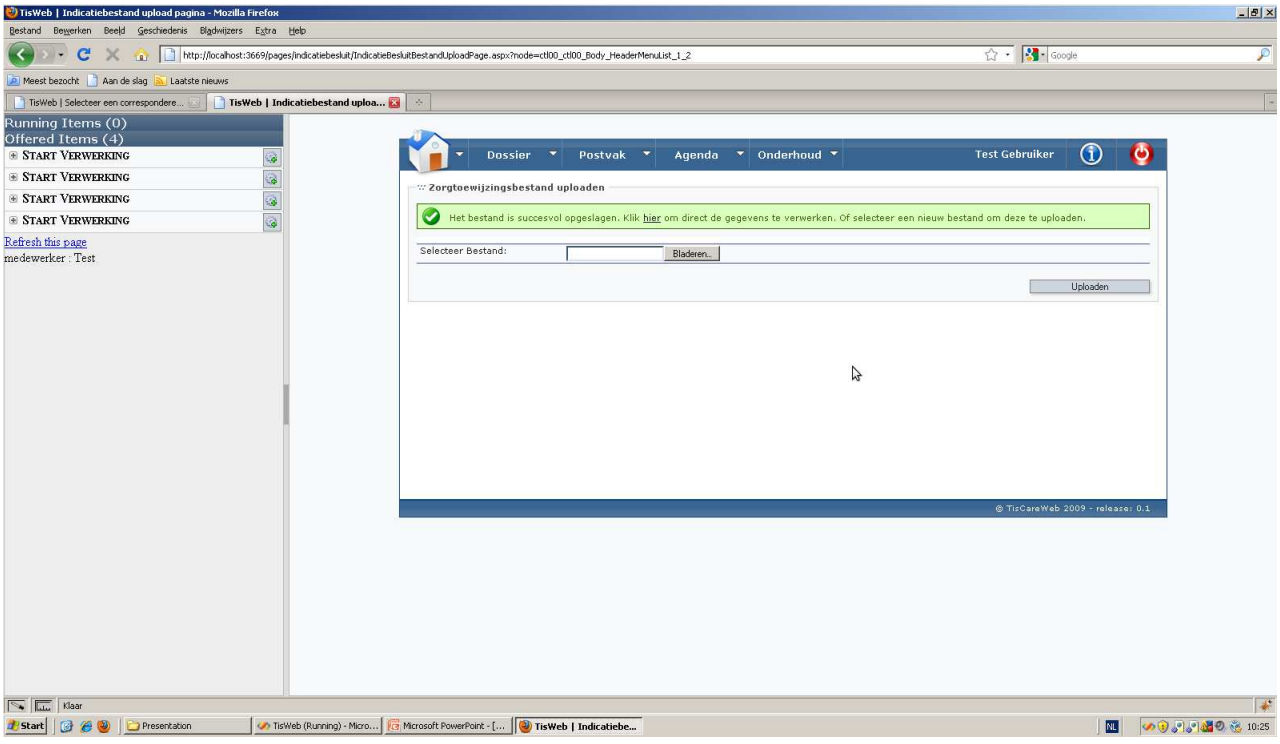

Figure A.3: Main screen and Task Manager after upload of *Indicatiebesluit*

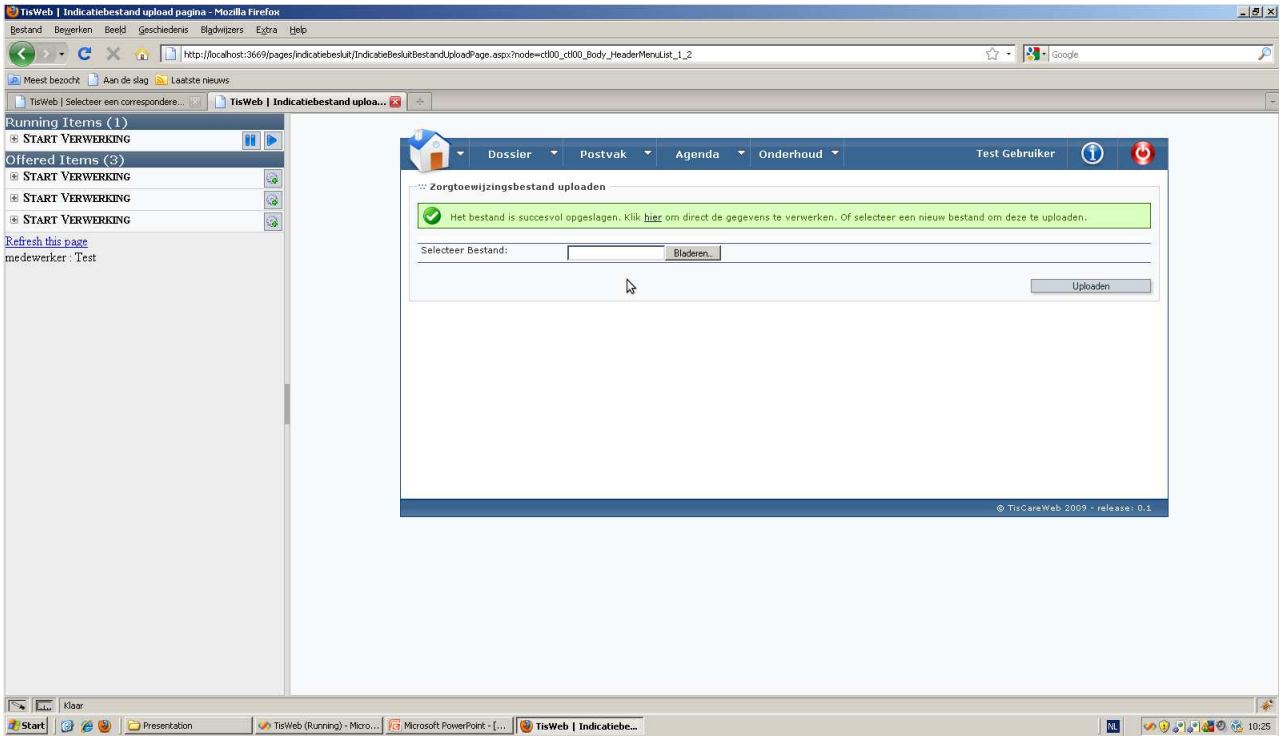

Figure A.4: Main screen and Task Manager after accepting a task

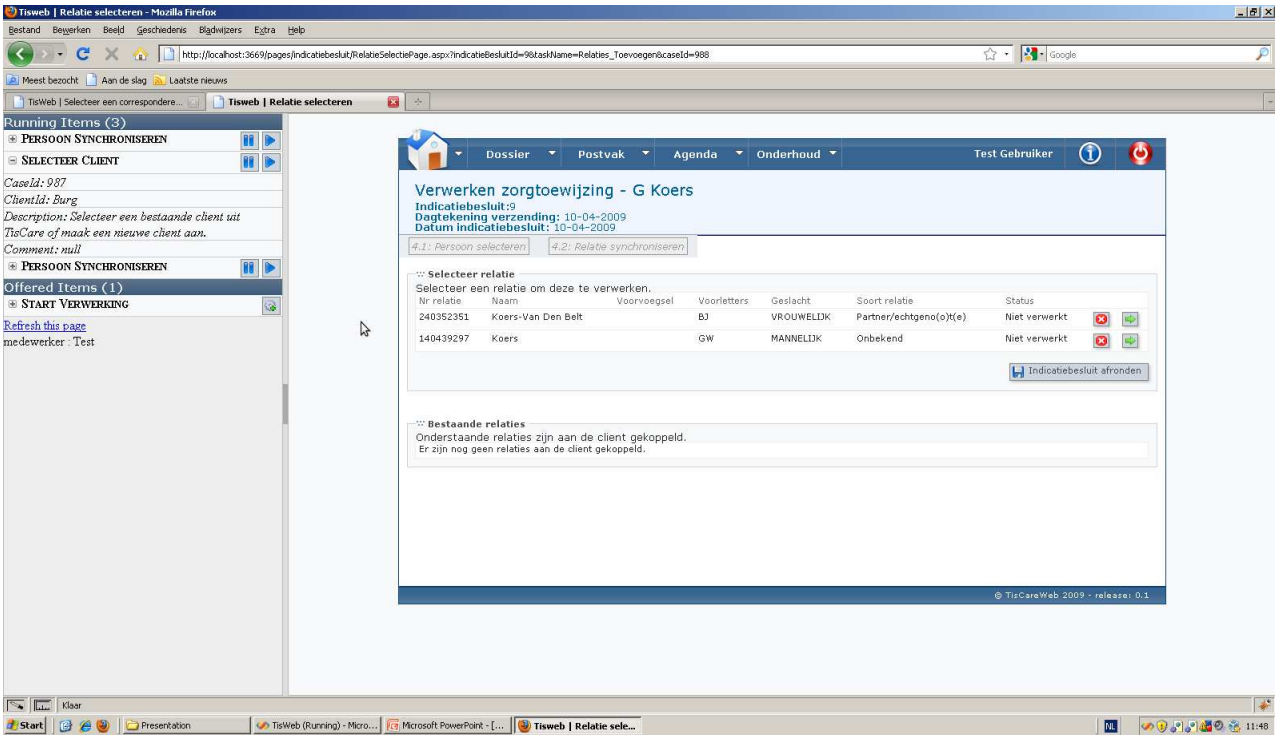

Figure A.5: Main screen and Task Manager with task details unfolded