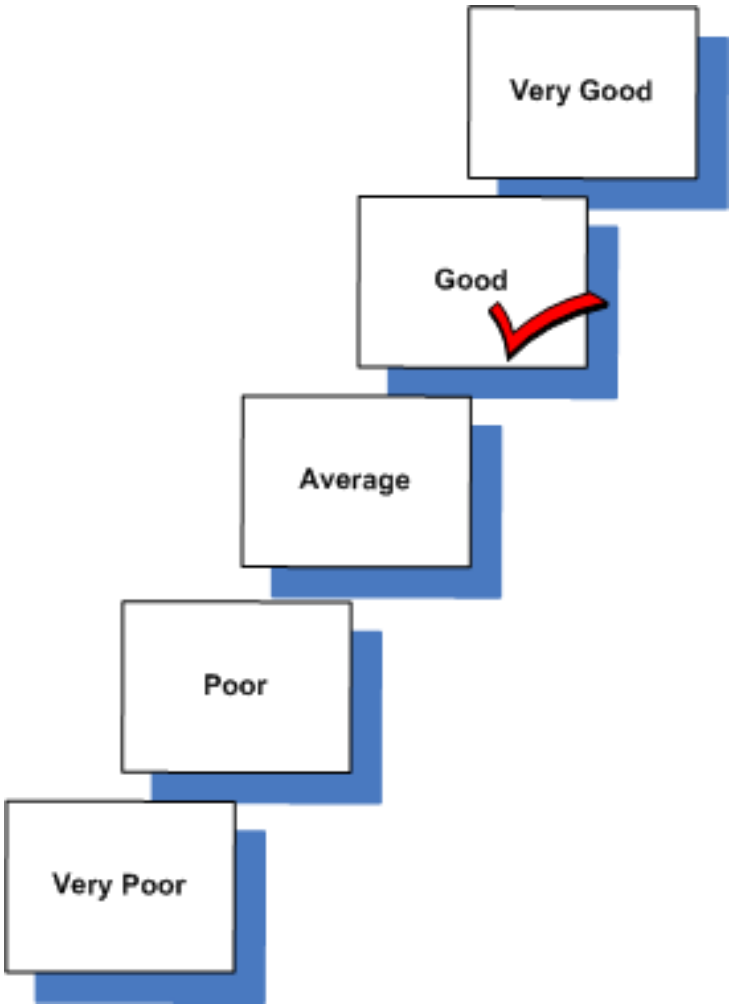
Towards a modeling tool evaluation method

A Radboud University Master Thesis - Information Science

Thesis number: 123 IK

Supervisor: Stijn Hoppenbrouwers

Author: Richard Willems - 0413410



19-03-2010

Abstract

In this thesis I will describe a new evaluation method, designed to evaluation the quality of a modeling tool, based on how well it supports the user's interactions. These interactions are based on the modeling language the tool is based on and the heuristics developed in human computer interaction. The research in this thesis was done using an extensive literature investigation using multiple books and articles, and interviews conducted at the Data Architectures & Metadata Management Group; a research group at the HAN applied sciences university. The resulting evaluation method was then tested using a case study conducted at the HAN applied sciences university during which several modeling sessions were observed and evaluated using TAP sessions. According to the research results, the evaluation method is promising, but there are still a few issues which need to be resolved.

Contents

Chapter 1

Introduction

1.1 General introduction

As a student of Information Science on the Radboud University Nijmegen you are required to write a final master thesis to finish your official master program. This report will be my master thesis which is supervised by Stijn Hoppenbrouwers, an assistant professor at the Radboud University Nijmegen. I am grateful to him for his assistance in finding an appropreate topic for this thesis and providing invaluable insights on the subject matter. I am also grateful to lector Guido Bakema from the Applied Sciences University of Arnhem and Nijmegen (HAN) who has kindly allowed me to work at his research department and consult him on various technical and organizatorial matters. All other members and employees from the Data Architectures & Metadata Management group were also very forthcoming in helping me complete this thesis.

1.2 Information on the Data Architectures & Metadata Management Group

The Data Architectures & Metadata Management group is a research group attached to the HAN University of Applied Sciences located in the Dutch city of Arnhem. For the last 15 years the HAN has played a major role in the research area of information system development based on conceptual modeling, automated model transformations and fully meta-driven tool support. (as described in [1]). To create one platform for these topics of research, the Data Architectures & Metadata Management group was created. Its research is to be used mainly in the world of higher education such as universities and organizations in both the Netherlands and abroad.

The Data Architectures & Metadata Management group also offers courses on various levels (as listed in [1]). First, there are the bachelor courses being given in the HAN computer science program having to do with information

system design and domain modeling. Second, they teach a complete Master program (in English) which is aimed at bachelor graduates from around the world. This Master program aims at developing advanced modeling techniques and skills. Third, evening courses are offered for employees from various companies. These courses also focus on modeling techniques and strategies but have more emphasis on skills rather then theory. Finally, the Data Architectures & Metadata Management group offer a minor (in English) called 'behind the screens' and are working on developing a minor called 'information systems development'.

The head of the group (called lector) is drs. Guido Bakema who is assisted by associate lector ir. Eddy Luursema. Chris Scholten MSc is responsible for the Master program and Jan-Pieter Zwart is head of promotion research into (and using) the FCO-IM modeling language. More information on the group may be found on www.han.nl/start/graduateschool/ onderzoek/lectoraten-kenniskringen/data-architectures-metadata-management. To contact the Data Architectures & Metadata Management group, mail may be sent to the group's secretary at lia.venhof@han.nl or at phone number 0031-(0)26-3658152.

1.3 Introduction to the thesis

In the world of system- and information modelling we see an enormous amount of tools and modelling techniques. One of these modelling techniques is FCO-IM, and the number one tool to work with FCO-IM is called CaseTalk. This tool has been developed in conjunction with Guido Bakema who is also one of the leading developers of FCO-IM. As a result of this cooperation we can clearly see that the way CaseTalk works is very similar to the way FCO-IM was actually designed, and that Bakema's way of thinking and modelling is reflected in both the interface and the workflow of CaseTalk. While at first this is seemingly a very good thing (after all, who would know better how to use FCO-IM then its developers) but from a HCI and modeling point of view I do not find this so obvious. No research has been done on how people actually think about modelling and how this way of thinking is reflected in the interface and (more importantly) the workflow of the program they are modelling with. This opinion is shared by Guido Bakema and Stijn Hoppenbrouwers who have decided to let me try and conceive a way of verifying to what extent a given modeling tool reflects the way people think about modeling in the way the tool allows the user to create models.

The end result of this thesis will be a (very early and primitive) model on how to measure a tool's ability to reflect the way people actually think about modelling while they are doing it and a trial investigation to verify the model and provide a proof-of-concept. The model itself will be developed from the FCO-IM/CaseTalk example. This means I will be using FCO-IM/CaseTalk as an instance of a modelling approach and try to generalize from FCO-IM/CaseTalk to modelling in general. However, the focus will be on FCO-IM/CaseTalk.

The evaluation model itself will be based on a combination of my own work

and insights, and the work done by [12] in the field of human-computer interaction. More specifically: I will use the heuristics developed by [12] and determine their effects on the modeling process itself. Then these heuristics will be analyzed and a method will be given which will allow the user to perform an evaluation of any modeling tool available and determine a grade (on a scale from 0-to-10) for this method. The resulting grade can later be used to perform a more extensive survey on the chosen tool, or to evaluate a number of tools to determine which one is the best for a chosen modeling language.

Finally, in order to demonstrate the use and effectiveness of the evaluation method I have developed, I will provide a case study in which I will use the method to perform a more extensive validation of CaseTalk. This case study will serve two goals. Firstly, it will allow me to perform a test run of the evaluation method. It will serve as a test to determine if it is possible to get results for the criteria I have devised given the fact that we are investigating modelers who may not have the required knowledge and expertise when it comes to the human computer interaction heuristics. Secondly, it will serve as an indication for the people from the Data Architectures & Metadata Management Group on how well CaseTalk is performing from a (for them) entirely new perspective: human computer interaction. An investigation from this perspective has not yet been performed on any version of CaseTalk (or its spiritual predecessors) which makes it an interesting take for the developers to reflect upon.

The general layout of this thesis is as follows: In chapter 2 I will provide an introduction to FCO-IM and CaseTalk, and provide an evaluation of CaseTalk from my perspective to introduce the evaluation method. In chapter 3 I will describe the method I have developed for tool-verification in a more general context. Then, in chapters 4 and 5 I will first describe the case-study done to verify the method and then provide the case study's results. Finally, in chapter 6 I will summarize my results and provide reference for possible future research to improve on my results.

1.4 Relevance

Recently, literature regarding modeling has focused more on the modeling process itself then it is still focusing on the end model. A good example is [14]. Even more recently, the concept of modeling through games is being explored in literature. Examples of this include [10] and [9]. These articles mostly suggest that the modeling process itself must be changed into a 'game' of some sorts to improve the ease-of-use, and make it more entertaining for the modeler to participate in the modeling process. Furthermore, two master thesis ([13] and [16]) have also explored the use of an actual game to assist in an organization's modeling problems and have actually provided a prototype of a modeling game.

Because of this, the question of evaluation a modeling process has become much more important. Without a valid evaluation method, it is hard to ascertain whether or not a certain modeling approach is better then the other. The work actually done in this area is extremely sparse. While looking for valid

literature about the subject I found some articles discussing the modeling process in general (such as [3]) but no work on evaluating the modeling process, or an actual study on the modeling process. The lack of proper research on this matter is especially strange if one looks on the vast number of modeling approaches available, a situation which is referred as the YAMA syndrome in [10]. Basically, the YAMA syndrome (Yet Another Modeling Approach) is a pun on the YAWL (Yet Another Workflow Language) acronym; which in turn is just a joke as well. The two subjects are related, but the joke is a general one. The authors in [10] are concerned by this YAMA syndrome from an operational perspective as it appears that everyone who is doing some modeling will design his or her own approach to it, and the lack of proper evaluation methods mean that those approached cannot be judged better or worse then the ones they want to replace.

Investigating the modeling process itself cannot be done without first looking at the tool with which the actual modeling takes place. Of course, there may still be some people who are doing most of their modeling using pen and paper, but if a model is to be used in a real context one is almost forced to digitalize the model in such a way that it may be interpreted by a computer. If the tool with which this is done does not suit the user's needs, then it would have a detrimental effect on the modeling process as a whole. To give an example: Let us assume that a certain modeling tool has an interface which forces the user to perform lots of interactions (such as clicking with a mouse, or typing) to even complete the simplest task. This may lead to the user skipping certain parts of the modeling process, or choosing a minimalist approach where this may have not been the most efficient course of action. The results on the actual model are obvious: it will not be as good as it might have been, purely due to the fact that the modeling tool used was not effective.

The lack of a proper evaluation method for such tools combined with the development of game modeling and the YAMA syndrome indicate that this thesis is relevant for the chosen field. Even if the method described in this paper is only in a very early stage it could still serve as the basis for more research into modeling process evaluation. Also, the case study will provide some examples of how people go about the modeling process which might serve as future reference when thinking about modeling process quality.

1.5 Research questions

To summarize what I have stated earlier, I have devised a two distinct research questions and related sub-questions. Those are:

1. How can we analyze a modeling tool from an HCI perspective?
    - What HCI perspective could be best used for analyzing a modeling tool?
    - What aspects from this perspective are relevant to the modeling process itself?

- How can we adapt this perspective and perform an evaluation of a modeling tool so that it results in a numerical value which may be further used in statistical research?
- How does the proposed evaluation method perform in a case study?

2. How can we use the evaluation method devised in question 1 to analyse CaseTalk?

   - How does modeling in FCO-IM work?
   - How is this modeling approach inplemented in CaseTalk?
   - What sort of case study on CaseTalk would yield the best results to use the evaluation method?

Please note that the research questions will be answered more or less in the order they are presented here, but to improve the readability of this thesis I have to discuss some of the sections dealing with the various research questions in another order. In chapter 6 I will list all the questions again and refer to the section that answers them.

1.6 Research method

To answer the questions posed in the previous section, I have used a two-fold research approach. First of all, I conducted a thorough a literature research to try and get an overview of the state-of-the-art, and to develop a deeper insight in the way the field of HCI looks at interactions. This research was conducted by reading literature from acknowledged scientific sources, and talking to experts on the fields relevant for this paper.

The second part of this thesis will be about a case study. I have conducted this research myself, by first setting up the experiment based on the conclusions of my literature research. A more complete overview may be found in chapter 4 The evaluation of the case study results was done from two perspectives: The first perspective was the one posed in the chapter 4 and had to do with the actual performance of CaseTalk. However, the second perspective (which is more relevant in this case) had to do with how well the case study itself performed. These conclusions may be found in chapter 6.

Chapter 2

FCO-IM and CaseTalk

In this chapter I will provide an extensive literature analysis on FCO-IM, CaseTalk and modeling in general. I will also provide an introduction on the evaluation method developed by myself which is based on the work of Nielsen [12]. FCO-IM will be explained step by step and all these steps will then be explained in the context of CaseTalk. This will provide an overview of how CaseTalk can be used to perform FCO-IM modeling. The evaluation of CaseTalk will be done (in this chapter) by myself to provide an introduction to the evaluation method. This method will be generalized in chapter 3 and will be used to further evaluate CaseTalk in chapters 4 and 5.

One note of caution: Both FCO-IM and its direct predecessor NIAM are Dutch research projects, and therefore a lot of documentation is originally in Dutch. This means that some of my sources are written in Dutch and direct citations from these sources will be in Dutch as well. All translations provided in this document are done by me and checked by Stijn Hoppenbrouwers. We are solely responsible for any errors in translation leading to misintepretation of the facts.

2.1 Introduction to FCO-IM

FCO-IM (or: Fully Communication Oriented Information Modeling) is best summarized by the authors themselves in [2], where they state their main principle to be: "FCO-IM beoogt niet de werklijkheid zelf te modelleren, maar de communicatie over de werkelijkheid." Or, to translate into English: "FCO-IM does not try to model reality itself, but rather communication about reality". This means that it tries to distantiate itself from the more traditional modeling languages which try to model reality based on 'raw' data such as fact types or object types (ORM, or UML) and instead focus on modeling the "ideas about reality". These ideas about reality will then lead back to the more traditional elements as mentioned earlier.

The first attempt to create such a modeling approach was done in the Nether-

lands and was called NIAM which stands for Natural language Information Analysis Method. This method was originally developed by G.M. Nijssen. In [4] the author explores NIAM in more detail and concludes that it is better to investigate facts about reality and derive reality itself from those facts. This was also concluded in [2]. One of the main conclusion from NIAM was that in order to facilitate 'representing reality through facts' it was necessary to develop 'representative examples' for reality and simply use natural language to represent them. Basically: sentence representation for facts. This is still what FCO-IM is based on.

But FCO-IM itself goes much further than NIAM did. NIAM simply collected the natural language sentences, distilled some fact-, object- and related types from it, and we were back at ORM with a different source. FCO-IM on the other hand also works on the sentences themselves. In fact, FCO-IM allows for the creation of a fully fledged information grammar which is modelled into the resulting end model.

The end model itself in FCO-IM is very similar to an ORM model, and anyone familiar with ORM will be able to recognize the FCO-IM model immediately and start interpreting what it shows. However, FCO-IM people decided to call the resulting diagram an 'information grammar diagram', or IGD for short. An example is shown in figure 2.1. In this illustration we can see both the similarity with ORM and the main differences. The biggest difference is noticable in the lettering near the object- and facttypes. Instead of simply showing the (possible) content of those types, they show the sentence (in plain English) which led to the creation of that type. These are called the 'factual sentences' and denoted by F*. Beneath those sentences you can see the example instances which lead to those sentences becoming fact types. Finally, the O* denote the objecttype-expressions which have been identified through the modeling process. More information on how this works can be found in section 2.2. .

As a final note I would like to point out another advantage of FCO-IM over most other modeling languages. One of the great challenges faced by anyone making models is the fact that the persons you are usually building them for (i.e. business people) don't have a clue on how to intepret models. Therefore, a lot of information regarding the model may be lost in translation, and therefore your customer (who is also your domain expert) has no way to validate the model you designed. With FCO-IM on the other hand, this problem is reduced. Since in your actual IGD the factual sentences are still represented, it is easier to understand for someone with no modeling experience. This means that the domain expert will be more capable in pointing out any factual errors you may have left in your model which leads in the end to a better model overall. This can be done with other modeling approaches as well, but these tend to come up with semi-formal sentences which might still be difficult to understand by your domain experts. FCO-IM still uses completely natural language sentences so this problem does not occur.

To summarize this section it is worthwhile to quote [2] who have come up with the principles of FCO-IM, and list them as follows: (translated from Dutch

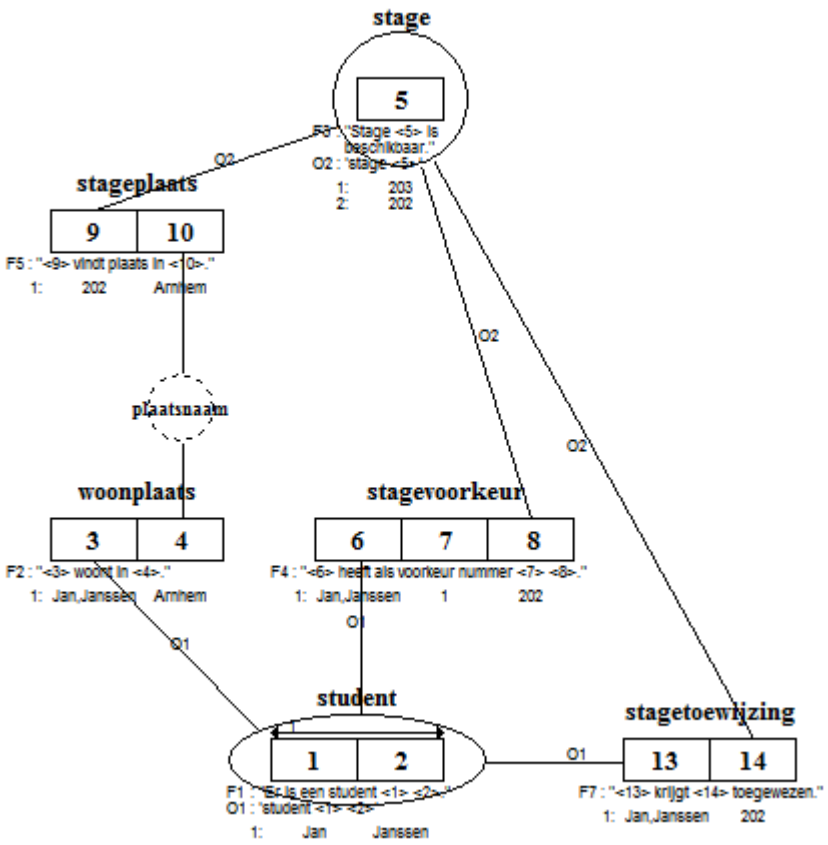


Figure 2.1: An example IGD

by myself)

- FCO-IM does not try to model reality itself, but rather the communication about reality
- FCO-IM has to model all aspects of communication which have to support the information system, but nothing more than that
- The domain expert has to be able to validate that all his communication is actually modelled in and FCO-IM information grammar
- FCO-IM information grammars and relational schemas have to be viewed with one and the same FCO-IM diagram technique. (the IGD standard)

2.2 modeling in FCO-IM

2.2.1 General Information modeling

To develop a method for evaluation the quality of an operational modeling process, we must first gain a deeper understanding of the modeling process itself. A general framework to describe the modeling process has been developed in Frederiks and van der Weide [3] which I will use as a basis. However, I will change the model in some aspects as I will explain in this section. This is due to the focus on FCO-IM for my method.

Basically, Frederiks and van der Weide [3] break the modeling process down into three distinct steps: elicitation, modeling and validation. These steps together form the modeling cycle. The first step, elicitation, is about gathering information from the domain. During this step, the domain experts and stakeholders have to be consulted to find all the relevant information. This step also involves identifying the universe of discourse and thereby narrowing down your domain as much as possible. Using the domain experts and stakeholders, the modellers have to summarize this information using natural language as much as possible. This is of course due to the fact that using natural language it is very easy to verify the information with the domain experts and stakeholder who are generally not very fluent in any formal language. A downside of course is that natural language tends to be ambiguous which might result in a situation where the domain experts and the modellers think they have come to an agreement, but are instead thinking about two entirely different implementations of what has been written down.

In Frederiks and van der Weide [3] this step has been identified as being singular. In other words: Once this step is completed it is not repeated in their model. This is due to the assumption that domain experts cannot speak the formal language used in the actual modeling process, and that therefore the domain experts can simply not play any significant role in verifying the model. However, given what we will see on FCO-IM in section 2.2 this does not necessarily hold true anymore. As modeling methods evolve and include natural language in the actual model, I propose that the elicitation step is included in

the entire cycle as even domain experts should be able to assist in validating the model itself. The end result may be seen in my modified overview in figure 2.2. As for the product of this step, this is not changed: The elicitation step should still end with a specification in natural language of the problem area.

To further make things clear, Frederiks and van der Weide [3] have defined three sub-steps within the elicitation process which should be followed in order. Those steps are: Collect significant information objects from the application domain, verbalize these information objects in a common language and reformulate the initial specification into a unifying format. I believe these steps are still valid, but another one should be added when the elicitation step is done in the modeling cycle: Verifying the model. I will place this step in figure 2.2 and with this step I mean that the stakeholder should (with the assistance of the modeler) examine the formal model and give comments on it which might (or might not) change the specification made in natural language.

The second step is called modeling. As the name suggests this is the main step of the modeling process. The definition given in Frederiks and van der Weide [3] is: "the intention of the modeling phase is to transform an informal specification into a formal specification". To be more precise: In the modeling step we try to transform the informal specification in natural language gained in the first step into a formal specification which adheres to the rules and regulations set for the chosen modeling method. During this process, the domain experts and stakeholders are of course not involved in any way. They simply lack the skills and training required for this step.

Again, Frederiks and van der Weide [3] have given us two sub steps to further define the modeling step: Discover significant modeling concepts (syntactical categories) and their relationships and match sentence structure on modeling concepts. Basically this mean the modeler has to Analise the information he is presented with and try and find categories in them under which a lot of the instances can be grouped. If we use FCO-IM as an example, we are looking for the elements in our sentences which are categories. It is in this step that the natural language ambiguity becomes dangerous: Concepts have to be found and given a meaning. If the informal specification was unclear or misunderstood by the stakeholders and domain experts we can end up with a model that does no longer reflect reality in such a way that the domain experts and stakeholders can agree with it.

The third step is validation. In Frederiks and van der Weide [3] this step is defined as being done completely by the modeler. He or she should generate phrases from the model and then verify these phrases using the informal specification, and then analyzing if the model still represents the reality as defined in the informal specification. However, as I have stated before I do not believe in this approach, and a modeling method like FCO-IM gives us the possibility to involve the stakeholders in the validation process as well. Therefore, I propose that as part of the validation step the domain experts and stakeholders should be involved in the validation process on two levels: First to see if the model reflects the reality of the informal specification. This should help to reduce ambiguity because of the natural language as well, since both parties are

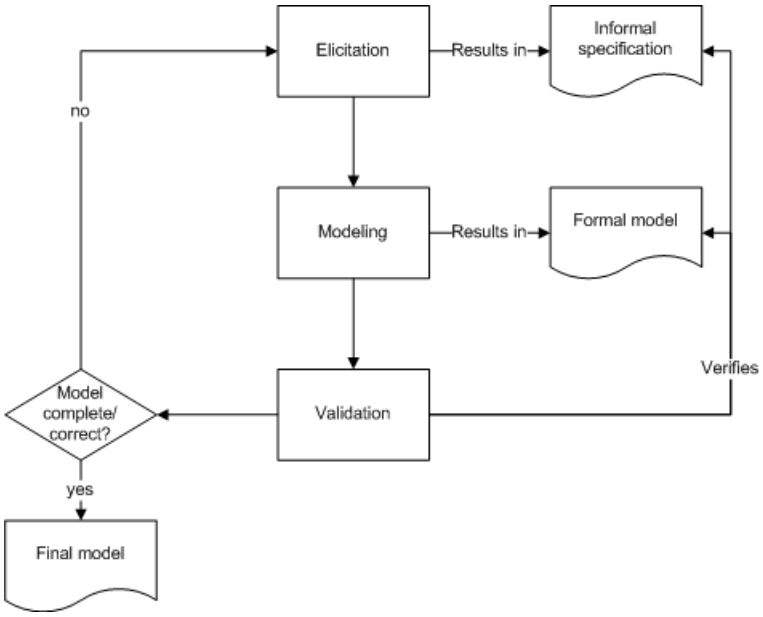


Figure 2.2: The modeling process, based on Frederiks and van der Weide [3]

involved. Second, the stakeholders and domain experts might have gained some new insights on what they want and do not want (because they were working on the model) which might lead them to change the informal specification as well. Again, this will be further clarified in figure 2.2.

Finally, after step three we can decide if the model is finished or not. If not, then the sequence should be repeated following the patterns laid out in figure 2.2. If so, then the model is in compliance with both the stakeholders and the modelers and it can be considered a finished product.

In figure 2.2 the modeling process is graphically explained. Please note that I have adapted the work of Frederiks and van der Weide [3] and changed it to meet my own criteria. As we can see, the process still starts at the elicitation step. This step results in the informal specification written in natural language. The second step is also still the modeling step, resulting in the formal model. Finally, the third step has been changed. I have called it the validation step (as explained earlier). During this step the modeler, together with the stakeholders and domain experts, analyzes the model and the informal specification to find out if the two are still in conjunction. If so, then the final model is delivered. If not, we go back to the elicitation step and start from there again.

2.2.2 Applying the Frederiks/van der Weide model to FCO-IM

As I have stated earlier, the main basis for the modeling process evaluation method will be the FCO-IM modeling approach. Therefore I will first apply the FCO-IM modeling approach, which will be described in great detail in section 2.2.3, to the adapted Frederiks/van der Weide model described in section 2.2.1. Please note that for people who are unfamilair with FCO-IM and even ORM in general, I advise to read section 2.2.3 first as it will discuss some of the terminology used in this modeling approach which is also relevant for this section. To continue, I will take the three steps identified in the adapted Frederiks/van der Weide model (AFW) and list the FCO-IM steps related to them. This will result in a clearer definition of the modeling process advocated in FCO-IM. In chapter 4 I will use these results to develop the evaluation method for the case study.

The first step in the AFW model is the elicitation phase. To reiterate: In this step we take the information from the domain (by using domain experts, stakeholders or literature research) and form a definition of the universe of discourse (the valid domain) in natural language. This we have called the 'informal model' in figure 2.2. In FCO-IM this step is exactly the same. Through discussions with the relevant people, the 'starting document' (see Bakema et al. [2] and section 2.2.3 for more information) is to be generated: A sort of informal model in natural language which is to be understood by both the domain experts and stakeholders, and the modelers who are going to develop the formal model. In FCO-IM this step can be considered even more important as it will also form the basis for the fact sentences and eventually the constraints.

The second AFW model step is the actual creation of the formal FCO-IM model. By this we mean everything from generating the LTL-fact-type expressions to applying constraints on the IGD. The methods for this process will be further defined in section 2.2.3, but I have summarized these steps in figure 2.3 . The parts where the domain experts and/or stakeholders can offer their input are defined in step three, as this is part of the validation and evaluation process. FCO-IM's end model is considered to be the actual IGD (with constraints), but the formal natural language model (containing the OTL-FTEs, the LTL-FTEs) can be considered a very important intermediary result and therefore I consider it to be an actual product of the FCO-IM modeling process. The fact sentences are not usually considered part of the FCO-IM model, as defined by Bakema et al. [2] and **?**].

The third and final AFW model step is the validation phase. In this phase the model is verified in two ways: Internaly and externaly. The internal validation is done by the modelers themselves. For FCO-IM this means that they go back to the informal specificiation and compare it to the formal language model and the IGD. They will first have to decide for themselves if the final IGD meets the specifications of the formal language model, and then in turn if the formal language model meets the specifications of the informal specification. When this is confirmed the domain experts and stakeholders will (together with the mod-

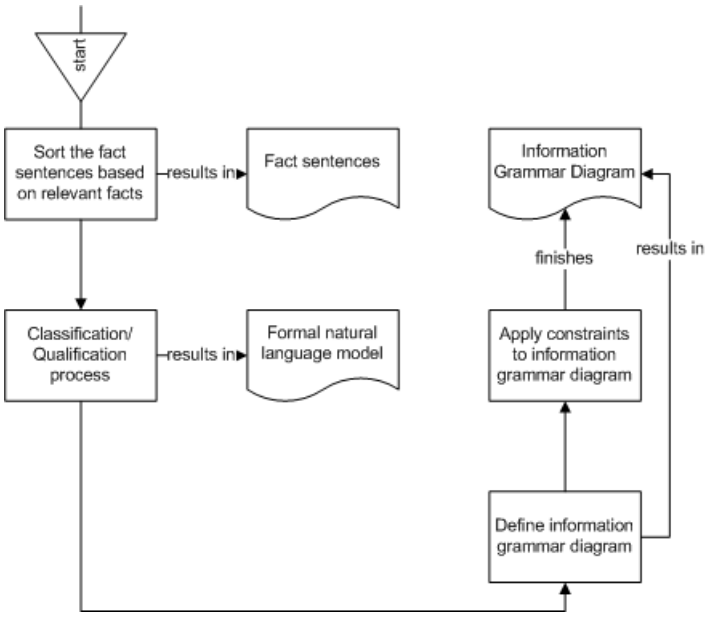


Figure 2.3: The FCO-IM modeling process and resulting data

elers) verify the same things, a step which we call the external validation. This means that again the IGD is discussed with them (and explained where needed) and compared to the formal natural language model. Because of FCO-IM's use of natural language, external validation is possible at all. In a fully technical or abstract model, external validation of the model itself is impossible due to the lack of understanding (generaly) by stakeholders and domain experts. Finally, the domain experts are once again asked to verify the informal specification as well.

When the final step is completed, the modelers will have to decide (in conjunction with the domain experts and/or stakeholders) if the model can be considered complete. If so, then the most recent model can be called the end result. If not, we go back to the elicitation step again and try to improve the model were necessary. Please note that in subsequent runs of the model, certain steps may be skipped if agreed upon by the modelers and domain experts.

2.2.3

Modeling in FCO-IM is not so much about modeling data (although in the end you will end up with it) but much more about modeling communication, as has also been made clear in section 2.1. The process of modeling in this fashion was first made (more or less) explicit with the work in [6], although this early method focussed more on actually modeling the facts which resulted from communication rather then the communication itself. So, communication

was a tool, not yet a goal. [6] were however very optimistic about the way interfaces and computer interaction would be modelled. In their work they claim that by the end of the 1990s computers which worked with formal natural language (what they call 'the fifth generation') would be comercially available. This has not come to pass, unfortunately, but their work was taken over and improved upon by the FCO-IM group. Using FCO-IM to model, you still are not able to use actual natural language as machine input, but it does give you the tools to strictly translate the natural language into a formal model which the computer can understand. Basically, an intermediary between the human and the machine.

Now before I explain the modeling method used in FCO-IM I would like to put forward a disclaimer: Everything explained from here on downward is actually the current view held by the designers of FCO-IM and the people who develop CaseTalk. This may or may not be the actual optimal method to model communication; that problem will explored in chapter 5 and 6. However, it is the basis from which I developed the research method presented in chapter 3, so it will be explained in some depth.

The first step to model communication through FCO-IM is obtaining a 'starting document' from your domain expert(s). This document should contain all relevant information and information processing processes. This document is written in natural language. The next step is to obtain instances for all the objects and lists mentioned in the starting document. These instances will later be used to develop the factual sentences so it is important to be thorough here.

The next step in the modeling process is the so-called classification/qualification phase (which may be abbreviated to ClaQua). The purpose of this phase is to find all the object-type level fact-type expressions, which is abbreviated to OTL-FTEs. From these OTL-FTEs we can derive the label-type level fact-fact type expressions which we shall abbreviate to LTL-FTEs. The combination of these two groups of expressions can be considered an intermediary result for the FCO-IM modeling process. The authors have not named this intermediary result so in the future I will refer to it as the 'formal natural language model'.

To start the ClaQua procedure we first have to go back to the fact sentences. The first step in the ClaQua procedure is to classify your facts. By classifying facts we mean identifying the fact sentences which refer to the same fact and grouping them together. For example, let us assume that we have four fact sentences:

- There is a criminal Jan Janssen
- There is a criminal Piet Pietersen
- There is a crime assault
- There is a crime murder

If we wanted to classify these four sentences we could identify two distinct categories. The first two sentences all refer to some criminal, while the third and fourth sentence refer to some crime. This means that these two groups

of sentences seem to indicate two distinct facts. Identifying these facts is the essence of this first qualificiation step. Notice however, that finding these fact categories might not be as easy as my example. The domain expert will most likely not provide the information so rigidly as I did. It is up to the modelers to find groupings which are both relevant and real.

Next, we need to identify our label type level fact type expressions or LTL-FTEs. LTL-FTEs are sentences stating facts in which the variables are identified and named. This is the qualification phase of the ClaQua approach. Bakema et al. [2] advise to use instances, remove the actuall instances and name the remaining open spots, but there is no real clearly defined method for getting from the instances to the LTL fact-type expressions. A LTL fact-type expression could look like this: "F1: '<first name><last name> has been found guilty of <crime>'". We see that the first- and last name of the person in question is a variable and so if the crime this person has committed. This fact-type is therefore about a committed crime. To skip ahead: This sentence is actually just a database structure which states that a certain crime event is identified by the person who did it and the crime which has been committed. Some concrete rules apply to what exactly the sentences should look like and what rules apply to writing them down, but it is beyond the scope of this thesis to discuss them in any great detail. For further information I would advise to read Bakema et al. [2] and **?**] (both in Dutch), both of which contain details and examples.

After the LTL fact type expressions are defined the ClaQua approach enters it's second phase. In this second phase we again classify and qualify, but this time we focus on finding any possible object-type expressions (OTEs) and label-types (LTs). These object-type expressions are specific parts of sentences which always identify the same object, and may consist of any combination of fixed parts and variables. It would be prudent however to only identify object-type expressions which occur more then once otherwise everything could be identified as objects. For example, it is logical to assume that a person will occur more then once. Therefore, we could write an object-type expression to say this. We would write it as: "O1: '<first name><last name>'". Now using this, we could re-write our F1 as: "F1: '<Person:O1> has been found guilty of <crime>'". Again, more rules apply here and I again refer you to the aforementioned literature.

Finaly, label-types (LTs) are where we keep the actual instances which were used in the fact sentences. So if we consider the example of a person, the label types which would make up a person could be called 'first name' and 'last name'. To summarize: Finding the OTEs and LTs is the classification phase, naming them is the qualification phase.

In order to illustrate the method mentioned above, consider the following example regarding criminals and their sentences. First we 'ask' the domain expert to provide some statements. That first phase has been left out. Next we summarize the statements according to their fact type and derive the first LTL fact-type expressions from them:

- *Fact sentence*

  - 'there is a criminal Jan Janssen'
  - 'there is a criminal Piet Pietersen'
  - 'there is a crime assault'
  - 'there is a crime murder'
  - 'Jan Janssen has been found guilty of the crime assault'
  - 'Piet Pietersen has been found guilty of the crime murder'
  - 'crime assault warrants 10 years in prison'
  - 'crime murder warrants 40 years in prison'

- *First LTL fact-type expressions*
  - F1: 'there is a criminal <first name> <last name>'
  - F2: 'there is a crime <crime category>'
  - F3: '<first name> <last name> has been found guilty of crime <crime category>'
  - F4: 'crime <crime category> warrants <prison time>'

Now from those first LTL fact-type expressions we derive the object-type expressions and then substitute them in the first set of LTL fact-type expressions. This will get us the object-type level fact-type expressions, or OTL-FTEs:

- OTL-FTEs
  - *OTEs*
    - O1: 'criminal <first name> <last name>'
    - O2: 'crime <crime category>'
  - *LTL fact-type expressions*
    - F1: 'there is a criminal <first name> <last name>'
    - F2: 'there is a crime <crime category>'
    - F3: '<Criminal:O1> has been found guilty of <Crime:O2>'
    - F4: '<Crime:O2> warrants <prison time>'

Once this step is completed and agree upon, the next step is to create an information grammar diagram (IGD). As stated and illustrated before, this diagram looks a lot like a regular ORM diagram, with the main difference that the information grammar is represented through the use of sentences. These sentences are positioned near the fact- and object-types for which they are relevant. The following example and its supporting illustrations assume that some knowledge of ORM is available. I will not discuss the details of ORM in this thesis. A very basic guide to ORM may be found at *http://www.orm.net/pdf/orm-emm98.pdf*.

To design the IGD a few things have to be considered. All OTEs become object-types. These circles are populated by the elements which are contained

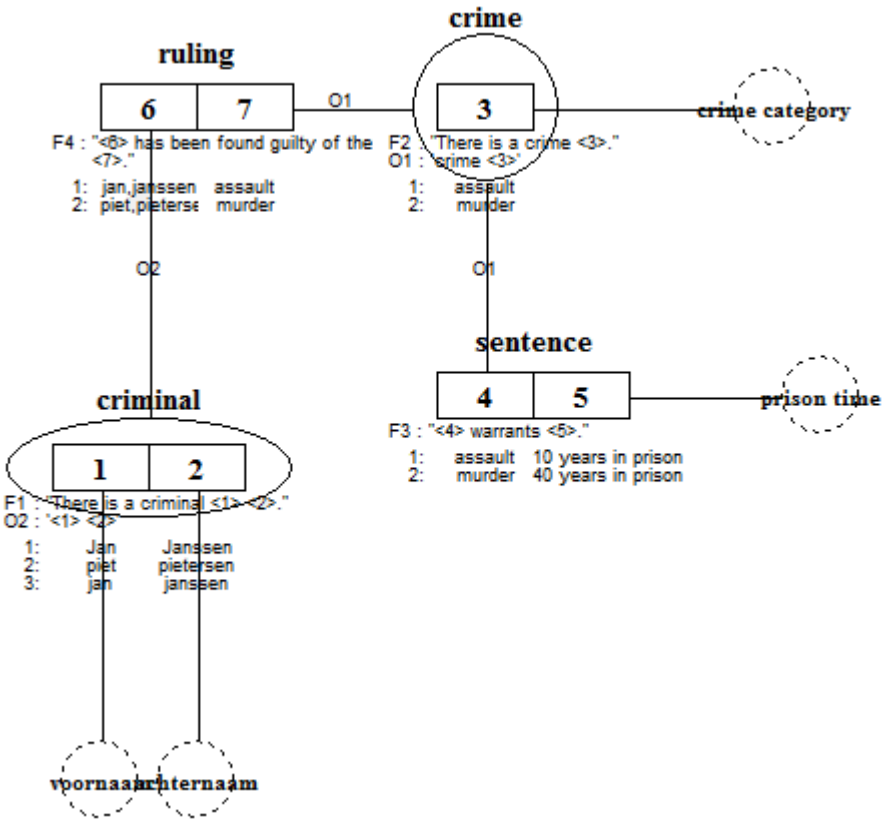


Figure 2.4: IGD from the 'crime' example

within the object-role type. In my example we can see that a 'criminal' is identified by his first- and last name. This means that the 'criminal' object-type will be created by joining a first- and last name as well. A 'crime' is constituted of just a crime category and therefore the object-type will also be built by just a crime category. The LTL fact-type expressions will become fact-types which are constituted by all the elements contained in the sentence. For example, F3 (the sentence concerning a judgment) will become a fact-type made by joining a criminal with a crime. All variables which are not specific object-types become label-types which contain the actual instances. The IGD derived from my example can be seen in figure 2.4.

2.2.4 Constraints in FCO-IM

The final step in creating a simple FCO-IM model is to identify and apply the constraints which are to be put on the IGD. Again; constraints used in a FCO-IM IGD are very similar to the constraints used by ORM. Constraints are generally needed to satisfy two demands: Make sure to model what the domain expert wants and prevent redundancy in your data. Without constraints, neither of those requirements may be fulfilled. For example, let us look at the 'criminal' example again. What is to prevent us from from creating another tuple like: 'crime assaults warrants 20 years in prison'. In the model, this means that we add 'assault' to label-type <4>, creating redundancy because it is already there and '20 years in prison' to label-type <5>. Now we have violated both the non-redundancy requirement and a domain requirement; the crime 'assault' now has two sentences! This must be prevented and it is there that we start to use constraints.

Constraints in FCO-IM are of course derived from domain expert. To find them we have to go back to the original design document again and analyze it again. Constraints are sentences in which conditions and rules are described. It is necessary for the modeller to gain some experience in actually finding the constraints as they may be hidden in the domain expert's information. FCO-IM offers little formal help to find constraints, and the procedures handed down by the designers also offer little information on how to actually find the constraints given the domain expert's design document. Discussions I had with FCO-IM's designers indicate that they consider this a separate field of requirements gathering and have therefore not invested a lot of time in finding a general approach that works well for FCO-IM. They have however, developed some procedures per constraint which I will refer to when discussion the constraint in question.

All constraints in FCO-IM are modelled onto the IGD, so the LTL fact-type expressions and the OTEs are not changed. The first commonly used constraint in FCO-IM IGD models is the so-called value constraint (VC). These constraints are applied to label-types and define the values which may be entered into this specific label type. For example, if we look at my example again we can see the label-type 'prison time'. It is logical to assume that this label type may not just contain anything. Prison sentences are probably in years (in my example, that is) and therefore we must reflect this fact in the IGD as well. In FCO-IM this constraint is applied by providing the set of available instances and putting it between brackets. To illustrate the 'prison-time' example (assuming all prison sentences are handed out in years and may range from 1 to 100 years in prison): {1...100}. The graphical representation can be found in figure 2.6. Please note that the number (1 in the example) just indicates the index of that constraint and has no bearing on the constraint itself.

The second constraint used in FCO-IM is the uniqueness constraint (UC). This constraint is applied to roles as opposed to the value constraint which is applied to label-types. The main purpose of this constraint is, as the name suggests, to force uniqueness on one or a combination of roles. For example, in the criminal example we can now have a crime with two or even more sentences

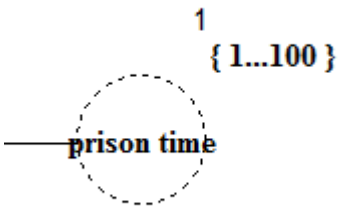


Figure 2.5: Graphical representation of a value constraints

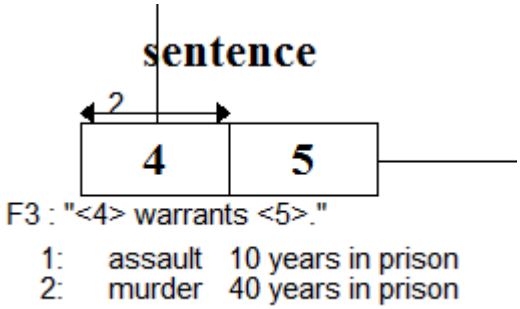


Figure 2.6: Graphical representation of a uniqueness constraint

attached to them. The domain expert will tell us that this is wrong, as a crime can only have one sentence. Therefore we could use the uniqueness constraint over role <4> which would force the user to only use every crime once. Now the fact statements 'crime murder warrants 40 years in prison' and 'crime murder warrants 50 years in prison' would be in violation of the constraint and made illegal in the model. On a side note: It is up to the actual implementation of course to further decide what to do with the illegal facts. FCO-IM has no clear guidelines on this. To visually denote this constraint in the IGD we put an arrow over the role (or roles) the UC concerns. Our example constraint in the IGD would look as depicted in figure 2.6. Again: The number is simply to label the constraint and means nothing for the constraint itself. To force a UC on a combination of roles one can simply draw an arrow over the multiple roles it is concerned with, or use the inter fact-type UC which is denoted as a line connecting the two roles with in the middle a circle containing the letter U for uniqueness. FCO-IM demands that each fact type has at least one uniqueness constraint present to prevent redundancy, as stated in Bakema et al. [2]. This is in line with the FCO-IM principle listed in section 2.1. FCO-IM's designers did put some procedures together to find and deploy the UCs for a given domain.

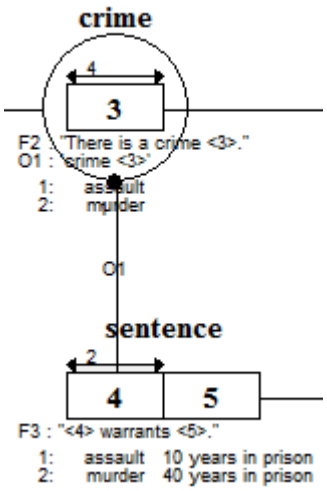


Figure 2.7: Graphical representation of a totality constraint

These can be found in Bakema et al. [2] sections 3.2 and 3.3..

I would like to introduce a critical remark here. FCO-IM's policy to force uniqueness constraints on at least one role in every fact type is based on implementation. In a database you want to have a unique identifier in every table to base the indexing on (primary key). This seems to be what FCO-IM does by demanding at least one UC. However, if we look at the model itself we can see that some fact types should not have a UC at all. The criminal fact-type for example. Why is a combination of first- and last name not allowed to occur more than once? Or just the first name? Or the last name? It is entirely conceivable to have two criminals with the same name. In that case, a non domain identifier (like a criminal identification number) could be introduced to solve the problem of identification, but the model should in my opinion not force it onto the modeller.

The next constraint I will discuss is the totality constraint (TC). This constraint is to indicate that all values in a given object-type have to appear in the fact-type they are connected to. For example, it is logical to assume that, unless you are living in some dictatorship, all crimes should have a sentence attached to it. To model this in the IGD we simply put a black dot at the line going from the object-type to the fact-type we want to put the totality constraint on. Figure 2.7 shows how this constraint is added to the IGD. TCs may be added to a role, but also to multiple roles. In that case we get a multiple role TC which is represented by a line between the roles with a circle in the middle which contains a black dot. In Bakema et al. [2] and **?**]you can find the procedures for systematically finding and defining TCs.

Finally, there are a number of smaller constraints in FCO-IM IGDs that I will

discuss only briefly, and will not illustrate with examples. The first constraint of this type if the subset constraint (SC). This constraint is used to indicate that the instances of a role (or set of roles) must be a subset of another role (or set of roles). In the diagram this is denoted as an arrow pointing from the role (or set of roles) which is to be the subset to the role (or set of roles) which has to contain the full set. The arrow branches when referring to more then one role. I find the use of this constraint to be debatable as it does not really seem to force anything at all. It merely illustrates a fact that just happens to be in the domain but this opinion is of course open to debate.

The second smaller constraint I will discuss is the equality constraint (EC). As the name suggests, an EC means that two roles (or set of roles) have to have the exact same set of values. In an IGD this can be denoted as a SC with a double arrowhead, or as a two TCs which would mean the same thing. ECs which are redundant are not drawn at all.

Finally, the last minor constraint is the cardinality constraint (CC). This constraint may be applied to a role or set of roles and indicates how many times a given value (or set of values) has to occur. This overrides the uniqueness constraint which may be seen as a CC with a value of one. In the IGD a CC is denoted as drawing a line from the role (or set of roles) it is relevant for and having this line and in a circle which contains '=n' where n is the value you wish to give to the CC.

All the aforementioned constraints fulfill the principles described in section 2.1 and allow for a clear and complete IGD to be formed out of the information presented by the domain experts. More information on techniques to verify this IGD and how to present it to the domain experts and stakeholders can be found in Bakema et al. [2] where a larger example and a complete overview of all the material discussed in sections 2.1 and 2.2 can be found as well.

2.2.5 Extra modeling aspects: Specialization and Generalization

One final aspect of FCO-IM modeling I wish to discuss is the matter of specialization and generalization. Although these two tools are not really needed to create a complete and functional IGD, they can be used to greatly clarify the IGD and even make it easier to work with it. Let us consider the example of a person to illustrate the issue. A person may be of gender male or female. By just putting this into a fact-type we can illustrate this fact, but if we want to continue working with this fact (so male or female only) it requires us to write a lot of constraints and derived fact- and object-types. It is easier to create a subtype which contains a subset of its supertype. In this case, the supertype would be 'persons' and the subtype could be 'male'. In this subtype you could put the population of 'persons' who are male, and continue working from this subtype as if it were an object-type itself. The population of this subtype could be derived by stating a fact-type 'male' and using that population to fill the subtype. The greatest advantage of this is that you can build an entire IGD for the male population without having to consider how you got it. This sim-

plifies everything from constraints to data gathering to eventually creating the relational schema out of it.

The second aspect, generalization, is of course the exact opposite of specialization. You take a couple of fact-types and create a 'fictional' object-type out of it. With fictional I mean that you just take a combination of facts which are not explicit in the universe of discourse, and generalize them into one object-type which can then be connected to other elements in the IGD. Again, generalization is to assist in making the resulting model as clean and clear as possible.

Specialization and generalization can always be avoided if so desired. This is stated in Bakema et al. [2], **?**] and Halpin and Nijssen [6]. They simply provide a way to help the modeller to make his or her model as clear as possible by providing tools to simplify the diagram. For FCO-IM, using specialization and generalization is recommended, but the resulting relational schema (and therefore, the resulting database) will be the same, whether specialization/generalization is used or not.

2.2.6 Beyond FCO-IM

When the IGD is complete in terms of domain information, there is another step which is recommended by the developers: Carry out tests to make sure the IGD only contains elementary fact-types. The resulting IGD will then be called an elementary IGD. From this IGD one can derive a relational database scheme which in turn is used to develop an actual database implementing the domain rules described in the IGD. The method used to do this is called the GLR algorithm which stands for grouping, lexicalizing and reducing. Describing this method is, however, beyond the scope of this thesis since I want to focus purely on the modeling itself. However, the GLR algorithm is a valuable component of the FCO-IM methodology since it allows the theoretical models to be put into practice, thereby greatly increasing the actual usefulness of FCO-IM in an organization. The relational database schema may in turn be converted into an actual database using SQL. This is also beyond the scope of this thesis. For now it is sufficient to know that this functionality exists and that it may be used to make FCO-IM useful in a real, organizational context.

2.2.7 FCO-IM versus ORM

In this final section on FCO-IM itself I would like to clarify a few differences between normal ORM and FCO-IM IGDs and why the two are not completely interchangeable. The biggest difference is in the use of object-types. In ORM pretty much anything can be an object-type, but more importantly: An object-type may exist by itself. In FCO-IM on the other hand, an object-type can only exist as an objectification (in ORM terminology) of fact-type. All other things are label-types. This means of course that an object-type in FCO-IM is guaranteed to actually mean something, and that it is much clearer what the object-type means. On the other hand, it forces the modeller to specify each and every object-type as a combination of facts which may make the model too

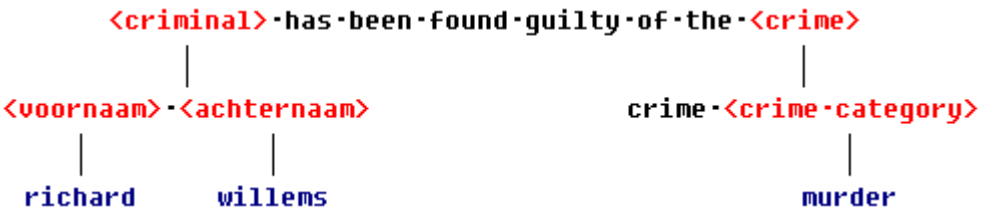


Figure 2.8: An example LTL expression in CaseTalk

complex. In Halpin [5] this is stated as well, but it goes on to say that ORM roles are relatively free and may be used throughout your diagram. However, we know that in FCO-IM all roles are indexed and are therefore rigidly maintained throughout the model. Again, this makes FCO-IM much clearer and allows for fewer mistakes, but it also makes modeling in FCO-IM harder since you constantly need to make sure you are using the right roles.

2.3 FCO-IM in CaseTalk

2.3.1 General analysis of CaseTalk

As I have already stated several times before, FCO-IM is implemented in the tool CaseTalk. CaseTalk itself was developed in conjunction with the developers of FCO-IM, and as such implements (nearly to the letter) the exact standards and specifications as laid out by those developers. This can be seen as both a blessing and a curse. As a blessing because this guarantees that FCO-IM is correctly implemented, and the theoretical literature may be applied perfectly when doing the modeling. However, it may also be seen as a curse since it means that there is little room for flexibility. Both the developers of CaseTalk and the modellers working with the tool are forced to follow the guidelines of the FCO-IM developers and are unable to put in any innovations of their own.

So let us now see how CaseTalk mirrors the modeling procedures laid out in sections 2.1 and 2.2. First of all, the program gets all its input from 'expressions'. Basically, these are the fact sentences we saw earlier. We can see that this fits the intentions of FCO-IM perfectly. Instead of using difficult wizards with a lot of 'technobabble', we can simply put in a sentence in natural language. Once the sentence is entered you are to identify what fact-type the sentence relates to. This can be done by either choosing that it is a new fact-type and typing the name or by selecting an already existing fact type from your model. The next step is to identify all elements in your expression. By doing this, you actually identify the LTL fact type expressions. These elements may be actual instances (label-types) or a role from another fact type. An example of how such an identification works can be found in figure 2.8.

When you enter the expressions, CaseTalk will automatically start identifying any possible objectification and separate them from the label-types which it

identifies as well. After you have finished entering the expressions, the next step is to draw the IGD. CaseTalk allows the user to select which parts of the IGD to draw by providing a drag-and-drop interface. Elements (such as fact-types, object-types and label-types) may be selected and drawn to the desired place on the canvas. CaseTalk automatically puts the elements which have connections together if so desired. This allows for the modeller to focus on specific parts of the model without the screen being cluttered by unnecessary elements. Also, the interface will try to present all elements as separate as possible to avoid cluttering and allow the user to line up everything as he or she sees fit.

It is also of interest to note that CaseTalk learns from patterns it sees. When an expression is entered based on a FTL which it has already learned it will automatically assign the correct labels to all the sentence elements and generate the appropriate LTL fact type expression. This allows the modeller to quickly add many expressions to the database which can later be used for model analysis as described in section 2.2 and in Bakema et al. [2].

The IGD viewing interface is also where constraints are applied in CaseTalk. As we have already seen in section 2.2.4 the constraints are put on the graphical representation of the IGD which is why it is represented in this fashion as well. Constraints may be applied in a number of ways, but the easiest way is to use the selection tool and select the constraint you want to put on your selected roles. Please note that CaseTalk follows the specific FCO-IM approach with regards to object-types and this is followed in the IGD viewer as well: Object-types cannot be selected at all. Instead, the roles making up the object will have to be selected. Only roles, fact-types and label-types may be selected and modified in the IGD viewer.

Once this step is completed CaseTalk is able to validate your model; and if so desired generate another model from it such as a relational schema. Validation is very accurate as it checks not only the constraints put on the model by FCO-IM's developers, but will also actually check if the population of the model matches the constraints put onto it by the user. This tool can help people to find the last bugs or errors in their model and help them correct those errors. Unfortunately, the error checker does not yet offer any suggestions on how the fix the errors, it merely states that they are there. When the validation is complete, the GLR algorithm may be called (explained earlier) to generate the relational schema which in turn can be turned into a variety of models; including an SQL generated database structure.

2.3.2 CaseTalk from a HCI perspective

In the previous section I have described the general outline of CaseTalk and how it works together with FCO-IM to form a modeling tool which is perfectly suited to the modeling language it is supposed to support. However, the above description was based on what we might call a workflow perspective. In this section, I will analyze CaseTalk again, but this time from an HCI perspective. This means I will look at the interactions the system offers to the user by way of use cases and provide an evaluation of those interactions based on a number

of criteria. The criteria I will use (for the HCI evaluation) are the heuristics defined by Nielsen [12]. These heuristics will be discussed later in this chapter.

I will start by defining use cases for CaseTalk. These use cases will not cover the entire application (since this would be too complex and certainly not entirely relevant) but they will focus on the modeling process itself. Also, some terminology used in the use cases is derived from FCO-IM standards. Therefore, in order to fully understand the use cases it is recommended to first read section 2.1. The use case definition I will use is based on Kulak and Guiney [11], and the resulting use cases will later be judged on the HCI criteria I mentioned earlier. Finally, the use cases will also form the basis for the case study interactions that I will observe. Please keep in mind, however, that I will change some use case elements with respect to Kulak and Guiney [11] to better suit my needs. The use case template I will use is explained later in this section.

Use cases are usually used for the requirements phase of a system that is still to be built. Therefore, it might be considered odd that I will use use cases to model the interactions of a system that already exists. However, I have very good reasons for this. First of all, use cases (in my opinion) are an easy way to convey information about interactions without having the completely specify them. If I would simply list all the possible interactions with the system and evaluate those based on the HCI criteria I would get a rather long list and might very well get lost in all the details. However, with use cases groups of interactions which are related to each other will be in one place (one use case) and I can easily evaluate those groups of interactions.

Secondly, the use cases will provide me with a good starting point to perform the case study. Since the use cases will describe a specific set of interactions with the system leading to one goal, I can assume that all of the user's actions during the case study will all be part of one or more use cases. So instead of having to classify each and every user interaction during the case study I can simply classify every group of interactions. This will lead to less overhead for me, and will allow the end results to be clearer and better defined.

The use case template I will use is the following:

- *name*: The name of the use case
  - *goal*: The goal which is to be achieved by this use case
  - *summary*: A short summary of the use case
  - *actor*: The actor who is performs this use case
  - *preconditions*: The conditions which have to be met before the use case may be run
  - *triggers*: The main event(s) that lead to the start of this use case
  - *BCOE*: The basic course of events; the interaction steps taken in this use case
  - *alternative paths*: If, why and how the use case may deviate from the BCOE

  - *postconditions*: The changes in system state that have occurred as a result of the execution of the use case

The use cases defined for CaseTalk are as follows:

- *name*: enter expression
  - *goal*: enter a new expression into the repository and store it
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    - a project must be open
  - *triggers*:
    - a new project is created
    - OR
    - modeler selects "new expression"
  - *BCOE*:
    - 1 - system prompts user for new expression
    - 2 - user enters expression
    - 3 - user qualifies the expression by giving either an object- or fact type name
    - OR
    - - user selects an already existing fact - or object type name
    - 4 - user presses the "qualify" button
    - 5 - system stores the expression
  - *alternative path*:
  - *postconditions*:
    - a new expression is entered into the repository

- *name*: qualify part of expression
  - *goal*: to qualify a certain part of a certain expression and store it
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    - an expression must have been entered into the system
  - *triggers*:
    - an expression is entered into the system

  - *BCOE*:
    * 1 - user selects part of the expression
    * 2 - user gives a name for this part
    * OR
    * - user selects an already existing fact- or object type name
    * 3 - user presses the "qualify" button
    * 4 - system stores the qualified part of the expression
  - *alternative path*:
  - *postconditions*:
    * part of an expression has been qualified

- *name*: confirm expression is qualified/quantified
  - *goal*: to inform the system an expression is completely qualified/quantified and store it
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    * an expression has at least one part qualified/quantified
  - *trigger*:
    * one part of an expression has been qualified/quantified
    * AND
    * user presses the "ready" button
  - *BCOE*:
    * 1 - system derives IGD elements from the qualification/quantification
  - *alternative path*:
  - *postconditions*:
    * IGD elements have been generated by the system

- *name*: confirm expression
  - *goal*: to confirm an expression has been correctly qualified/quantified
  - *summary*:
  - *actors*: modeler
  - *preconditions*:

    - an expression must have been qualified/quantified on all level
  - *triggers*:
    - an entire expression has been qualified/quantified on all levels
    - AND
    - user presses the "ok" button
  - *BCOE*:
    - 1 - the system stores both the expression and the generated IGD elements
  - *alternative path*:
  - *postconditions*:
    - the expression is stored
    - the IGD elements are stored

- *name*: undo ClaQua step
  - *goal*: to undo one step taken in the ClaQua process
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    - the user must have taken at least one step in the ClaQua process
  - *triggers*:
    - user presses the "undo" button during the expression session
  - *BCOE*:
    - 1 - the system undos the last step taken in the ClaQua process
  - *alternative path*:
  - *postconditions*:
    - the system is back at the situation it was in before the last ClaQua step was taken

---

- *name*: cancel expression
  - *goal*: to cancel the entire expression process
  - *summary*:
  - *actors*: modeler
  - *preconditions*:

    - an expression session must be opened
  - *triggers*:
    - user presses the "cancel" button during the expression session
  - *BCOE*:
    - 1 - the system cancels the expression operation and deletes all data
  - *alternative path*:
  - *postconditions*:
    - the system is back to the main screen

- *name*: delete repository item
  - *goal*: to delete one specific item from the respository
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    - at least one repository item has been previously generated
  - *triggers*:
    - user selects a repository item
    - AND
    - user presses CTRL+DEL
  - *BCOE*:
    - 1 - user selects a repository item
    - 2 - uses presses CTRL+DEL on the keyboard
    - 3 - system prompts for a confirmation and displays which associated repository items will be deleted with it
    - 4 - the user confirms the delete operation
    - 5 - the system deletes the respository item and its associated repository items
  - *alternative path*:
    - 4 - the user cancels the delete operation
  - *postconditions*:
    - one repository item and its associated respository items are deleted from the repository

---

- *name*: generate new IGD
  - *goal*: to generate a new (and blank) IGD
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
  - *triggers*:
    - user selects "new diagram"
  - *BCOE*:
    - 1 - system generates a new IGD diagram
    - 2 - system displays new IGD diagram
  - *alternative path*:
  - *postconditions*:
    - a new IGD diagram is stored into the system

- *name*: add respository item to IGD
  - *goal*: to add one specific repository item to the currently active and opened IGD
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    - at least one repository item has been previously generated
  - *triggers*:
    - user drags a repository item
  - *BCOE*:
    - 1 - user selects a repository item
    - 2 - user drags the selected repository item to the IGD currently in view
    - 3 - the system adds the repository item to the IGD
    - 4 - the system identifies all possible connections to already existing items in the IGD and displays these
    - 5 - the system will store the diagram
  - *alternative path*:
  - *postconditions*:
    - a new repository item is now added to the IGD

- *name*: delete respository item from IGD
  - *goal*: to delete one or more repository items from the IGD
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    - at least one repository item has been previously added to the IGD
  - *triggers*:
    - user selects one or more IGD items
    - AND
    - user presses DEL
  - *BCOE*:
    - 1 - user selects one or more IGD items
    - 2 - user presses the DEL button on the keyboard
    - 3 - the system deletes the item(s) from the IGD
    - 4 - the system identified all possible connections to the deleted items and deletes these as well
  - *alternative path*:
  - *postconditions*:
    - one or more repository items are now deleted from the IGD

- *name*: add unicity constraint
  - *goal*: to add a unicity constraint to one role, or a combination of roles
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    - at least one fact type has been generated in the IGD
  - *triggers*:
    - user selects one or more roles
    - AND
    - user presses the 'add unicity constraint' button
  - *BCOE*:

    - 1 - user selects one or more roles
    - 2 - user presses the 'add unicity constraint' button
    - 3 - the system adds the unicity constraint to the repository
    - 4 - the system adds the unicity constraint to the IGD
  - *alternative path*:
  - *postconditions*:
    - a new unicity constraint is added to the model

- *name*: delete unicity constraint
  - *goal*: to delete a unicity constraint to one role, or a combination of roles
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    - at least one fact type has been generated in the IGD
    - AND
    - one fact type has at least one unicity constraint
  - *triggers*:
    - user selects a unicity constraint
    - AND
    - user selects the 'delete constraint' option from the menu
  - *BCOE*:
    - 1 - user selects one unicity constraint
    - 2 - user selects the 'delete constraint' option from the context menu
    - 3 - the system deletes the unicity constraint
    - 4 - the systemremoves the unicity constraint from the IGD
  - *alternative path*:
  - *postconditions*:
    - one unicity constraint is removed from the model

- *name*: add totality constraint
  - *goal*: to add a totality constraint to a role / object type combination
  - *summary*:

  - *actors*: modeler
  - *preconditions*:
    * at least one fact type has been generated in the IGD
    * AND
    * this fact type is connected to an object type
  - *triggers*:
    * user selects one or more roles
    * AND
    * user presses the 'add totality constraint' button
  - *BCOE*:
    * 1 - user selects one or more roles which are connected to an object type
    * 2 - user presses the 'add totality constraint' button
    * 3 - the system adds the totality constraint to the repository
    * 4 - the system adds the totality constraint to the IGD
  - *alternative path*:
  - *postconditions*:
    * a new totality constraint is added to the model

- *name*: delete totality constraint
  - *goal*: to delete a totality constraint from one role
  - *summary*:
  - *actors*: modeler
  - *preconditions*:
    * at least one fact type has been generated in the IGD
    * AND
    * one fact type has at least one totality constraint
  - *triggers*:
    * user selects a totality constraint
    * AND
    * user selects the 'delete constraint' option from the menu
  - *BCOE*:
    * 1 - user selects one totality constraint
    * 2 - user selects the 'delete constraint' option from the context menu

    - 3 - the system deletes the totality constraint
    - 4 - the systemremoves the totality constraint from the IGD
- *alternative path*:
- *postconditions*:
    - one totality constraint is removed from the model

2.4 Analyzing CaseTalk

In the previous session I identified all of CaseTalk's main interactions (with respect to the modeling process) by way of use cases. In this section I will perform a heuristics evaluation of CaseTalk based on those interactions. The basis of this evaluation will be the work done by Jakob Nielsen; who is considered to be an expert in the field of usability. His heuristics are listed in Nielsen [12], and I will start by taking those heuristics and applying them to CaseTalk where applicable. Not all of his heuristics will be usable because of the distinct focus of my work. Others may not be relevant. This is why not all the heuristics in Nielsen [12] are used in this research. The reasoning behind exclusion or inclusion of a particular heuristic will be explained in later in this section.

This specific heuristics evaluation will be not be supported by any statistical data. Rather, it is my opinion on these heuristics backed by arguments and illustrated by examples from CaseTalk itself. It is not the main goal of this thesis to perform a heuristic evaluation of CaseTalk (it will serve as a field test of the evaluation method), but the data gathered here will help in evaluating the modeling process as will be explained in chapter 3.

2.4.1 Evaluating the heuristics

Heuristic 1 - Visibility of system status

This heuristic is defined by Nielsen [12] as *'The system should always keep users informed about what is going on, through appropriate feedback within reasonable time'*. This first heuristic is a very interesting one straight away. Let us first take a look at the definition of system status. In the article, this definition is not made clear and subsequent research in related articles have not provided me with a very clear and unambiguous definition. After some discussion with Stijn Hoppenbrouwers I have decided that this heuristic is probably best applied to the status of *what* the user is trying to achieve with the application. In this case, this is related to creating the model. So the system status would be some sort of indicator on how far the modeling process has proceeded. A similar system is being used for modeling in the world of 'modeling as games' which I briefly referred to earlier, and which is discussed in the works of Schotten [13], Hoppenbrouwers et al. [10] and Wilmont [16]. If we look at the modeling process then, the relevance of this heuristic becomes very obvious. If the system can evaluate the progress of the user's work, and actually provide feedback on the

quality and progress of that work it would be a tremendous help for the user to ascertain the quality and correctness of his work.

However, because this heuristic is only derived from a very new field, it is not surprising to see that CaseTalk does not have this feature at all. However, it is a very safe bet to assume that practically no other commercially available modeling tool will have any such functionality, since it is still something that is still merely being explored academically, and has not yet been seen in any main stream applications. Therefore, I have to conclude that although this heuristic is certainly very relevant to the modeling process, I will not yet consider it to be a criterium for the evaluation model because of the fact that its implementation has not yet been attempted in any real main stream modeling tool. In the future however, this decision may be changed depending on whether or not the idea of objective based program design advocated in the 'modeling as a game' literature gains a real foothold.

Heuristic 2 - Match between system and the real world

The definition of this heuristic is given in Nielsen [12] as *'The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in natural and logical order'.* This heuristic is of course very important to investigate in the context of the modeling process. It allows us to examine how well the modeling process is translated into the online tool at the modeler's disposal (in this case: CaseTalk).

As I have already described in section 2.3 the tool was developed in conjunction with the people who actually designed FCO-IM. Because of this, the terminology in CaseTalk is not just similar to the terminology used in FCO-IM but is exactly the same. The same goes for all other criteria set out in this heuristic. If one is able to create a FCO-IM model using pen-and-paper, one is also able to create this model using CaseTalk. All methods, terms and graphical representations are exactly the same and therefore this heuristic has been followed perfectly.

Heuristic 3 - User control and freedom

Nielsen [12] describes this heuristic as *'Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo'.* The relation between this heuristic and the modeling process is very clear. To make it easier for the modelers to correct mistakes an extended undo and redo functionality should be available. If for example the modeler made a mistake in classifying an expression CaseTalk should not force him to completely repeat the process for classifying all over again. Therefore, the extent to which this heuristic is followed has a profound impact on the modeling process.

If we look at CaseTalk we can see that this heuristic has in fact been clearly followed. Especially when looking at the classification/qualification process we

can see that there is always an option to undo one of your previous actions without having to cancel and start all over again. When we look at the process of drawing the IGD and applying constraints on it we also see a lot of functionality in this area. Actions, or parts of actions, can always be undone without losing any other work. You will only lose the work you actually want to undo.

Heuristic 4 - Consistency and standards

In Nielsen [12] the definition of this heuristic states: *'Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform conventions'.* Again, this heuristic is important for the modeling process. Consistency of the program will help in creating consistent models. If the program itself was inconsistent it could result in models being inconsistent because of the modeler being unsure on how to model his ideas into CaseTalk.

CaseTalk follows this heuristic perfectly. The program remains consistent throughout the modeling process and all operations and actions continue to yield the same sort of results. Also, the program is laid out according to a standard 'Windows-based' layout which is familiar to anyone who has ever worked with a graphical operating system.

Heuristic 5 - Error prevention

This heuristic is defined by Nielsen [12] as: *'Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a real confirmation option before they commit to the action'.* This heuristic is relevant to investigate because of the fact that when in CaseTalk you are looking at an error, you are in fact also looking at an error in the actual modeling process itself (if we disregard actual system errors for the moment). If CaseTalk allows for good error prevention and detection, the modeling process itself would also benefit.

Unfortunately, this heuristic is not completely adhered to in CaseTalk. There are instances in which the modeler is allowed to classify/qualify expressions with spelling errors. Because CaseTalk does not understand the syntax of natural language it will simply assume that the expression means something else entirely and therefore create a new fact type expression. In no way does CaseTalk inform the user that the expression is extremely similar to another one, yet slightly different. This means that errors can be made when the modeler translates his own mental modeling process into CaseTalk which in the long term might lead to faulty models.

Heuristic 6 - Recognition rather than recall

In Nielsen [12] this heuristic is defined as: *'Minimize the user's memory load by making objects, actions and options visible. The user should not have to remember information from one part of the dialog to another. Instructions for*

use of the system should be visible or easily retrievable whenever appropriate'. I consider this heuristic to be important, but not very important. This specific heuristic is more about interface design, and the speed with which the user can accomplish certain tasks within the interface. If we look at the impact this heuristic can have on the modeling process itself I can only conclude that the only impact it will have is on the speed of translating the mental model in the modeler's head to an actual digital model on the computer. So it has no impact on the quality of the model, but rather on the speed at which the modeler can actually deliver the model.

CaseTalk has followed this heuristic quite nicely. The most relevant source of information and details is the repository in which all elements are located (such as object types, fact types, label types, etc). This repository is (unless changed by the user) always visible in the bottom-left corner of the screen. All other views are also easily accessed through the project-manager in the top-left corner. So all data is made available throughout the program where-ever possible.

The second part of this heuristic (about the system instructions) is poorly followed. There is little to no help except for a tutorial and the actual online tool tips are not very helpful and in some cases even placed near the wrong items. Furthermore, the manual itself assumes a lot of knowledge about CaseTalk. This is probably a deliberate decision made by the developers, but in terms of making the program itself accessible this might not have been the perfect choice.

Heuristic 7 - Flexibility and efficiency of use

Nielsen [12] defines this heuristic as: *'Accelerators - unseen by the novice user - may often speed up the interaction for the expert user such that the system can cater to both the inexperienced and experienced users. Allows user to tailor frequent actions'.* For this heuristic I can pretty much conclude the same thing as I did for heuristic 6: Following it may increase the speed with which the modeler can produce his models, but has very little influence on the quality of the model itself.

As far as CaseTalk is concerned: this heuristic is not very well followed. All the wizards are just that: wizards. There is no way for an experienced user to shorten or improve those wizards as they are laid out by the programmers. Furthermore, aside from positioning the various windows, CaseTalk's interface can not be changed at all in terms of functionality. Therefore, the CaseTalk tool is a bit rigid and inflexible.

Heuristic 8 - Aesthetic and minimalist design

This heuristic is defined by Nielsen [12] as: *'Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in dialogue competes with the relevant units of information and diminishes their relative visibility'.* Again a rather important heuristic. In order for the modeler to focus on the modeling process, it is important that there is not too much

clutter on the screen. If this heuristic is followed then this will be the case. If there is too much information on the screen the modeler's train of thought might run the risk of being derailed and the mental model not adequately translated to an actual digital model in CaseTalk.

CaseTalk is in many ways very minimalistic when it comes to providing information. From that we seem to be able to conclude that this heuristic was followed. However, the context menus used in CaseTalk tend to hold too much information and can cause confusion for the modeler working with the program which can be confusing. Especially when working with constraints, CaseTalk offers information which may not be completely relevant or even understandable to the average user. Therefore, the conclusion would have to be that this heuristic is not very well followed which might cause confusion for the modeler.

Heuristic 9 - Help users recognize, diagnose and recover from errors

The definition of this heuristic according to 12is: *'Error messages should be expressed in plain language (no codes), precisely indicate the problem and constructively suggest a solution'.* This heuristic is a bit ambiguous. What is meant by 'error messages'? Is the author referring to errors in the program, or does he mean errors with respect to the functionality of the program? If we assume the first possible answer, then this heuristic is not very relevant for a modeler. If the program crashes, it crashes, and no amount of error messages will help the modeler to recover his lost model or help him in any way to get it back faster. However, if we assume the second answer then we do see some relevance. Error messages might indicate actual problems in the model itself such as an inconsistency in the instances. Since this is more relevant for this thesis I will for now assume the second meaning: error messages with respect to the actual content.

In CaseTalk we do not see a lot of error messages concerning the actual modeling. As I have already mentioned before, CaseTalk will just about accept anything when it comes to instances and expressions. Only when you transform the IGD into some sort of relational model (which is outside the scope of this thesis) will it start checking instances and such. So in terms of the heuristic, CaseTalk does very little to help users recognize errors, and virtually nothing to help them recover from the errors they made. To summarize: This heuristic (using my interpretation of it) is not followed in CaseTalk.

Heuristic 10 - Help and documentation

12 defines this heuristic as: *'Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large'.* This heuristic does not seem relevant here. The documentation on how to work with CaseTalk is not done by the programmers, but consists of all the literature regarding FCO-IM.

All that is left for the programmers of CaseTalk is to provide short instructions on how to translate the FCO-IM modeling principles into the program. This has been done fairly well through the use of some tutorials, but it is not strictly relevant to evaluate CaseTalk.

That being said, it is perhaps interesting to mention that the documentation in CaseTalk which is actually ABOUT CaseTalk (and how to work with it) is rather minimal. The help file contains nothing more than a few tutorials and does not cover all functionality. There is more help available online, but that means it is not readily available and is also not easily accessible from the point in the program where you are working (i.e. you can not simply press F1 to get relevant help).

2.4.2 Conclusion based on the heuristics

From the 10 heuristics defined in 12 I have identified 8 which are relevant to the modeling process itself. This has been explained in the previous section. In this section I will perform a final evaluation of CaseTalk based on the separate conclusions drawn on the relevant heuristics. This conclusion will help me in mapping and evaluating the modeling process of FCO-IM as a whole. If the tool used to create a model is found to be lacking, then we can already assume that the modeling process as a whole might be lacking in this area too. If for example the tool does not support the modeler through feedback, error checking and the like the quality of the model would probably decrease as well. Please bear in mind that this is an assumption. I have not been able to find previous research which relates the quality of the tool to the quality of the model, but 9 and 14 seem to suggest that there is a positive relation between the two as well.

To summarize things, I have first created a short overview of the heuristic evaluation done in the previous section. This can be found in table 2.1. The evaluation is done in a 5-step scale: {- -, -, o, +, + +}. Ranging from very poorly (- -) to very good (+ +). To perform the final evaluation I need to quantify these results, so I have decided to assign numerical values to the evaluation marks. A (- -) will deduce 2 points, a (-) will deduce 1 point, a (o) will not yield any points, a (+) will gain 1 point, and finally the (++) will gain 2 points. This means that in theory, the program could (provided there are 8 scored heuristics) gain anything from negative 16 to 16 points. To further make the assessment more clear I will change this to a 0-to-10 scale by adding 16 points to the total; dividing the result by 32 and multiplying it by 10. The results of this calculation can be seen in table 2.2.

However, the scores we have just calculated are still based on usability heuristics, whereas we are much more interested in facilitating the modeling process. As we can see in table 2.2, CaseTalk does not score very high in terms of usability. However, some heuristics are more important for the modeling process than others, so let us start by identifying which of the 8 selected heuristics are extra important when it comes to facilitating the modeling process.

Looking at the list of heuristics I have selected four which I feel are strongly related to facilitating the modeling process when looking at CaseTalk. These

Heuristic name	Evaluation
2. Match between system and the real world	+ +
3. User control and freedom	+ +
4. Consistency and standards	+ +
5. Error prevention	-
6. Recognition rather than recall	o
7. Flexibility and efficiency of use	- -
8. Aesthetic and minimalist design	-
9. Help users recognize, diagnose and recover from errors	-

Table 2.1: Summary of heuristic analysis

Total score	Absolute score	Corrected score
+1	17	5,3125

Table 2.2: Heuristic scores - 0-to-10

heuristics are 2, 3, 4 and 5. Heuristic 2 is about matching the real world. The importance of this heuristic is obvious: If the program does not adhere to the rules and regulations which apply to the modeling language it tries to facilitate, then the models created with this tool will not be of any good quality since they will not adhere to the rules and regulations as well.

Heuristic 3 is about user control and freedom. This is also important because it allows for the modeler to quickly change things or correct mistakes. If this heuristic is followed, then the modeler will be able to correct mistakes with only very little effort which is not only very pleasant for the modeler, but it will also prevent mistakes! If for example the modeler would have to retrace many steps to correct one single mistake there is always the chance that in the process of correcting the mistake, new mistakes will be made which will further reduce the quality of the modeling process and therefore: the model itself.

The next heuristic is number 4. This heuristic is about being consistent and following standards. Especially the first part makes this very important for the modeling process. If the modeling tool is consistent throughout the modeling process then all modeling steps will be very clear to the modeler. If on the other hand the tool is not consistent then there is a probability that the modeler will not recognize this completely and start making mistakes in the modeling process which in turn lead to bad models. A consistent tool means the modeler is always clear on what he is doing which in turn leads to fewer mistakes and in the end: a better model.

The final heuristic is number 5, which has to do with error prevention. Error prevention of course is crucial when doing modeling work. If you are trying to model something which is inconsistent, illogical or just plain wrong you would like the tool you are using to provide some feedback and ask you if that last action was truly what you wanted to do. With this feature present in your tool, the modeling process will become less error prone, and in the end lead to a

Heuristic name	Weight	Score
2. Match between system and the real world	h	+ + + +
3. User control and freedom	h	+ + + +
4. Consistency and standards	h	+ + + +
5. Error prevention	h	- -
6. Recognition rather than recall	n	o
7. Flexibility and efficiency of use	n	- -
8. Aesthetic and minimalist design	n	-
9. Help users recognize, diagnose and recover from errors	n	-

Table 2.3: Modified heuristics analysis

Total score	Absolute score	Corrected score
+6	30	6,25

Table 2.4: Modified heuristic score - 0-to-10

better overall model. When it is not present, there is the chance of the modeler making mistakes he did not intend to make (i.e. just made an error inputting the model) increases, and the overall quality of the final model decreases because of this.

To implement those high priority heuristics into my scoring model, I have decided to double the weight of scoring on those four items. For the scoring chart, this means that there are 8 extra points to gain, and 8 extra points to lose. The scoring table has been changed to reflect this, and the result is shown in table 2.3. Again we translate these scores to a 0-to-10 scale. This time we add 24 to the final score, divide all results by 48 and we multiply the answer by 10 again. The final result of this calculation is shown in table 2.4.

From these scores we can conclude that if we put some emphasis on the heuristics which are important to the modeling process, CaseTalk seems to be doing a fairly good job. Therefore I conclude that CaseTalk has a positive effect on the modeling process when using FCO-IM. In chapter 3 I will generalize the method I have just described which means it can be used to test and evaluate other modeling tools as well. Please note that the result for CaseTalk as such is still not very informative since there is no comparative baseline. However, if more tools were to be evaluated in a similar fashion, we could start a comparison and see which tool (when analyzed using HCI usability heuristics) serves the modeling process best.

As an addition to what I have stated in this section I would like to discuss heuristic 1 one more time. Since I already stated that this heuristic may be added to the method in the future, I would like to quickly explain how I envision this in the current model. Heuristic 1 would also be considered of high importance; the same as heuristics 2, 3, 4 and 5. This means that the score for this heuristic also has to be doubled in accordance to the method I described earlier. Also, the total scoring system would have to be adjusted to allow for the

plus or minus 4 points that could be gained from this heuristic. I will explain this procedure in slightly more detail in chapter 3.

2.5 Chapter conclusions

In this chapter I have identified a few key elements of CaseTalk, which in chapter 4 will be used to develop the case study. These are:

- A basic model of all modeling steps in FCO-IM.
- The use cases (interactions) for CaseTalk. These are all the interactions which are relevant to the modeling process.
- The heuristics based on Nielsen [12] which are relevant for the modeling process.
- An evaluation method for modeling tools, with the current focus on CaseTalk.

Chapter 3

General Evaluation Method

In this chapter I will generalize the evaluation method first described and worked out in section 2.3.2. This means I will aim to make it applicable to most, if not all modeling tools and provide some insight on how to gather the scores for the various heuristics used in this method.

3.1 Introduction to the Problem

A lot of work on modeling on evaluating the quality of models in the past. Most of this work has focused on the quality of models *produced*, but a more interesting area has of yet not seen a lot of attention: The quality of the modeling process itself [10]. This is probably due to the fact that it is mostly the end result which matters most. It is this end result which is used to define businesses, IT systems or databases so I agree with the fact that it is important to check the quality of this end result. But in my opinion (which is shared by [9] and [10]) the end result is still very much dependent on the process of actually getting it, so it is relevant to do more research in this area. As I have stated before, the focus of my work in this thesis is evaluating the modeling tool, which in turn is part of the modeling process. I will therefore not actually contribute to the field of model process evaluation itself, but instead focus on creating an evaluation method for a small part of the modeling process; the modeling tool and its usage.

To tackle this problem, it is interesting to look at the field of human-computer interaction (HCI). In this field we can see that a lot of work is being done on how to analyse the way people work with computers and how they do the things that they do. This is opposed to for example computer science where the main focus is on how the program itself works; and if it follows the rules laid out during the program's design. A very interesting take on using HCI to evaluate CaseTalk was already provided in section 2.3.2.

And this is of course interesting for the problem of evaluating the modeling process. We can draw the parallel between what HCI does for computer

programs and modeling: They both focus on the same problem area: how do people do it? So if this is true, we should be able to modify and then use the procedures from HCI to evaluate modeling tools as well. This problem has been partly explored in work done by [13] and [16] and I will try to expand their work and devise a general method which I will then test in practice by performing a case study on CaseTalk, the FCO-IM tool which is explained in depth in section 2.3.

3.2 Evaluation Method

The general method for evaluating a modeling tool using HCI heuristics has already been partially described and applied to CaseTalk in section 2.3.2. The main focus of this method are the 10 heuristics defined by [12] of which I have choosen eight which I consider relevant to the modeling process when seen from the user's perspective. Of these eight heuristics I have identified four which I consider to be of a higher importance than the other ones. When these heuristics are given scores, a 'grade' can be calculated. This grade, ranging from 0 to 10, can be used to compare it to modeling tools for the same modeling language, or even to compare various modeling languages and their tools to eachother. To summarize my conclusions from section 2.3.2, and to provide an overview which is a little bit more general than the one provided earlier I have compiled a list which will be given below. This list only contains the remaining heuristics and provides a short description from a modeler's perspective of every heuristic. The description will also provide information on how to score that specific heuristic. Also, in table 3.1 a summary of the heuristic's importance may be found.

3.2.1 Evaluation Criteria

1. *Match between system and the real world* - This heuristic must be judged based on how well the tool actually implements the modeling language. Are the same rules followed? Is the modeler able to perform all the actions he would be able to perform with pen and paper? Do the models look the same in the tool as they would look on paper? When there is a significant difference between the modeling tool and the actual modeling language then this heuristic should not receive a very high score. This heuristic has been given a high importance because it has a real effect on the resulting model itself.

2. *User control and freedom* - This heuristic must be judged based on how easy it is for the user to make mistakes (while experimenting for example) and correcting those mistakes once they are identified. Is there a simple 'undo' step or does the user have to go back many steps and complete already completed steps all over again? If it is easy to correct mistakes this heuristic should receive a high score. If correcting mistakes means that previous work could or will be lost, then a low score should be given.

This heuristic is given a high importance because it has a very big effect on the efficiency of the modeling process.

3. *Consistency and standards* - This heuristic must be judged based on how consistent the modeling tools presents itself, and how well it adheres to set standards in the industry. Especially the first part is relevant here. If the tool is not consistent, it could throw the modeler off-balance and result in models that are not what the modeler had intended. For example, if a button which should apply a certain constraint to an aspect of the model does so differently every time you have another item selected then this can be very confusing. If many if these examples exist in the modeling tool then the heuristic should be scored low. If however, it is perfectly clear what every bit of functionality does anywhere and anytime in the program, then a high score should be given to this heuristic. This heuristic is given a high importance because its implementation has a direct effect on the resulting model itself.

4. *Error prevention* - This heuristic must be judged based on how well the modeling tools helps the user in preventing errors in the model itself. A modeling tool could for example give warnings when the user tries to develop something which is inconsistent with other aspects of the model. Or in graphical terms, prevent the user from drawing models which are not allowed by the modeling language itself. More intelligent programs could (by use of AI, pattern recognition and the like) even assist the user by preventing semantically errors in the program. If the tool is capable of helping the modeler with this, then this heuristic should be given a high score. If, however, the program simply accepts all input by the user and does not check validity in any way then it should receive a low score. This heuristic is given a high importance because any errors in the model of course have a detrimental effect on its quality.

5. *Recognition rather than recall* - This heuristic should be judged based on how well the tool presents the relevant information to the user. If the user needs to remember a lot of information just by memory (because it is poorly placed) it might lead to errors in the model since the user might not remember how he named a certain element, or what constraint was put on which model element. If the program smartly displays all relevant information onscreen for the user it will save the modeler time that would otherwise be spent in looking up previous aspects of the model. When it is very easy for the user to recall previous work (or even better: when it is constantly displayed onscreen) this heuristic should receive a high score. When a lot of work is required to find previous work then it should get a low score. This heuristic is given a normal importance since its main effect will be on the speed of the modeling process, not its quality.

6. *Flexibility and efficiency of use* - This heuristic should be judged based on how well the user can modify the program to suit his needs. If the user has

for example a preferred way of modeling certain things, then he should be able to adapt the interface of the program to reflect this method. When the interface is more flexible, the modeler will be able to work faster. If the interface is customizable then this heuristic should receive a high score. If not, then a low score should be administered. This heuristic is given a normal importance since its main effect will be on the speed of the modeling process, not is quality.

7. *Aesthetic and minimalist design* - This heuristic should be judged based on how 'clean' the tool's interface looks. When working, a modeler should not be distracted by too many items and features on screen. In fact, only the required elements for his current, specific modeling action should be visible. This will help the modeler focus on his current task and will have a positive effect on the quality of models. Too much clutter might also lead to a slower modeling process, since the modeler would be spending too much time on finding the required functionality in the interface. If the interface is nice and clean, this heuristic should be given a high score. However, if there is much cluttering going on and too many flashy things are present in the interface without a good reason then a low score should be given. This heuristic is given normal importance since it is mostly concerned with the speed of the modeling process, and only has a very small effect on the actually quality of the resulting model.

8. *Help users recognize, diagnose and recover from errors* - This heuristic should be judged based on how well the tool helps users to find errors in their models, and correct those errors. The tool (since it 'knows' the modeling language) should be able to identify any syntactical errors (and perhaps even semantics, in the near future) in the model and present these to the modeler. Ideally, the tool should also come up with possible fixes to assist the modeler in correcting his model. If the tool has many such features then this heuristic should receive a high score. However, should the tool not be able to detect errors at all, then a low score should be given. This heuristic is given normal importance, as it is only a secondary safety feature and it only adds to the correctness of the final model, not the actual quality of it. If semantic errors can be identified as well, then this heuristic should be given a higher importance, but since this is not yet commercially available I have chosen to ignore it for now.

Heuristic Name	Importance
Match between system and the real world	High
User control and freedom	High
Consistency and standards	High
Error prevention	High
Recognition rather than recall	Normal
Flexibility and efficiency of use	Normal
Aesthetic and minimalist design	Normal
Help users recognize, diagnose, and recover from errors	Normal

Table 3.1: Heuristics importance

3.2.2 Scoring

In section 3.2.1 I discussed what criteria should be scored. In this section, I will discuss how to score these heuristics and how to derive a 'grade' from this scoring system. Please note that I am not yet discussing the aspect of *how* to get these scores; this will be discussed in a later section as it is a rather big issue in the evaluation method.

The scoring itself is done on an ordinal scale [7]. I could have scored it on a nominal scale (i.e. give it a score from 0-to-10), but I have decided not to for a couple of reasons. First of all, nominal scales are hard to score. It is very easy to leave the scorer wondering whether to go for a 6 or a 7. This means that the scoring might become vague in some points, which leads to murky results. An ordinal scale is usually more limited (less options to choose from), and when chosen properly it will also have more clearly defined scoring points. A good example of this includes the well known scale ranging from strongly disagree to strongly agree. I have chosen a similar scoring system for my evaluation method as well.

My second reason for using an ordinal scale is that it allows for a much better expression of what someone 'feels'. If we look at a nominal scale we can only see numbers. Of course, we know or might be told that every number higher then five indicates that we like something, but it is a rather hard concept to express. With an ordinal scale on the other hand we can actually choose the labels for the scores which could be used to make the answers more domain specific, which in turn leads back to argument one: the scoring will be clearer for whoever has to decide on the score.

Another good reason for using an ordinal scale is that it can be transformed. As I will explain later, I have identified two ways to get scores for the heuristics used in my evaluation method. Because of this, it might make more sense to use different labels for the different ways of getting data. When the amount of answers you are using is clear and you have assigned values to the index of the answer the labels do not matter anymore. So instead of using a (strongly disagree) to (strongly agree) type of scale, you might replace it with a scale ranging from (- -) to (+ +). The values assigned to the answer stays the same,

Scale 1	Scale 2	Standard value	High importance value
strongly disagree	- -	-2	-4
disagree	-	-1	-2
neutral/no opinion	o	0	0
agree	+	+1	+2
strongly agree	+ +	+2	+4

Table 3.2: Scoring examples

Total score	Grade	Total score	Grade	Total score	Grade
-12	0	-3	3.75	6	7.50
-11	0.42	-2	4.17	7	7.92
-10	0.83	-1	4.58	8	8.33
-9	1.25	0	5.00	9	8.75
-8	1.67	1	5.42	10	9.17
-7	2.08	2	5.83	11	9.58
-6	2.50	3	6.25	12	10.00
-5	2.92	4	6.67		
-4	3.33	5	7.08		

Table 3.3: Scoring table

but the first type of label might be more applicable to a certain research method like survey, whereas the second type of label might be more applicable to another type of research method, like a think aloud session.

So now that it is clear *why* I used this sort of scaling, let us now look at *how* I used the ordinal scale. To simplify actually getting the results, I have decided on a 5-grade scale. In the example given in section 2.3.2 I used a scale ranging from (- -) to (+ +), with (o) as neutral. All of these were assigned numerical values as well, ranging from (-2) to (+2). The scoring is done per heuristic. When a heuristic is deemed *high importance* (as defined in section 3.2.1) the scoring is doubled by the investigator, not the scorer himself. This is to provide one uniform scoring criterion for the scorer and reduce ambiguity in the scoring itself. By doubling the score I mean simply doubling the value. For example, if the score was (-), meaning (-1), it would be doubled to (- -), or (-2). If the score was (+ +), meaning (+2), the investigator would double the score to (+ + + +), or (+4). The neutral value (o), stay the same.

As I have said before, the actual labels of the scores may be changed. So instead of using a scale from (- -) to (+ +), one could use a scale from (strongly disagree) to (strongly agree). This would especially be useful when performing a survey to get a tool's scores. The first scale is more recommended when performing a think aloud session. How to actually use these two methods will be explained in section 3.3. A few examples are given in table 3.2.

Once the scores per heuristic are known, a few simple mathematical steps can be applied to translate the final score to a 0-to-10 system. If one adds up the

scores from the 8 heuristics (keeping in mind which ones are of high importance and which are not) you arrive at the final score for the modeling tool you are investigating. This can be translated to a grade by looking in table 3.3. The grade may then be used to perform any sort of statistical analysis, since we have now translated it into an interval scale [7] which may be used for these purposes.

In summary, I have made use of the properties of an ordinal scale in order to make the scoring process easier for both the scorer and the investigator. Then I translated these ordinal results into an interval scale so that the results can be used in statistical analyses.

3.3 Getting the Scores

Now that I have explained on what criteria the heuristics should be judged, and how the scoring should be done, I will now move on the explaining how to actually get the scores for the heuristics. Talking to various people, I have concluded that there are basically two distinct ways of getting the required information: a survey, and a think aloud session. Both have their distinct pros and cons, and I will discuss them both to some detail in this section. Please note that I do not think one is generally better then the other, but I in specific situation a certain preference might be observed and justified.

3.3.1 Performing a Survey

Performing a survey is one way to get the scores for the different heuristics. Doing a survey requires basically two things: A good survey, and a large enough group to perform the survey on [7]. I will briefly discuss both these aspects, but for more information I refer you to the relevant literature as I am not an expert on these things. However, the size of the group has to be carefully considered. If we look at our intended target population (people using the tool you are investigating) we have to realize that looking at this group and comparing to the world population we have to realize that it is actually not so big a group. Therefore, it is important to make sure to get a big enough response. There are many methods for getting a good response which are discussed in various pieces of literature. I just want to state here, that getting a large group for a survey is not as easy as it seems. Even if you manage to contact a large enough group, there is no guarantee that you will get a 100% response so you will have to do a lot of pre-research to ensure a sizable enough sample population.

My evaluation method is not fixed to any particular 'level of depth' in the program. Up until now I have discussed only the possibility of investigating the modeling tool as a whole. However, because of the fact that the heuristics by [12] can be applied not only to a program as a whole, but also on parts of the program, we can do the same with the survey. This leads to two approaches.

The basic idea of performing a survey to get scores for the various heuristics is to create a questionnaire. In this questionnaire one can simply state the heuristics with their appropriate description and ask the respondent to select

their score for this heuristic. It would seem most logical to use the (strongly disagree) to (strongly agree) scale for such a survey since the other scale might not feel natural to your sample population.

A more advanced possibility to perform the survey is to look at the interaction patterns identified in section 2.3.2. Instead of directly asking about the heuristics, the researcher could score all interaction patterns seperately, and thus gain a grade for every interaction. The average of these grades would then be the final grade for the tool he is evaluating. Of course, not all modeling tools have the same interaction patterns as CaseTalk, so when such a survey is preferred it is necessary for the researcher to identify the interaction pattern in his specific modeling tool first. The advantage of this method is that you would get a better average for the tool as a total since you are no longer asking the participant to find an average for the entire program but instead offer him the chance the give his opinion on little bits of the program. Also, since you are getting results per interaction it would give you another way to statistically compare data between researches.

The most obvious downside for performing an advanced survey is that it requires a lot more work, both for the researcher and the participant. This is widely considered to be bad, especially for the problem of participation [7]. The more complex a survey becomes, the less likely it is to get people to participate in your survey since it requires more of their time. Therefore, it is important to consider your participation group before deciding on how complex you want to make your survey.

3.3.2 Performing a Think Aloud Session

The next possibility is performing a think aloud session. Such a session is useful because it offers a lot more information then just performing a survey. [7] and [8] both conclude that a survey will give you a large amount of information, but the information is actually very shallow. In other words, it will only give you answers you were specifically looking for, and nothing else. In a think aloud session on the other hand, you can get much more information which might also be relevant to draw other conclusions from then you had originally anticipated.

I will not discuss in any great detail how exactly a think aloud session (TAP session) works. Since it is a widely used concept a lot of information is available on the internet, and I find it beyond the scope of my thesis to further go into this matter. To summarize a TAP session: A person is asked to complete a certain task, for example to produce a model. Furthermore, this participant is asked to verbalize the thoughts he is having on a certain subject, generally the subject you are investigating. These verbalizations are recorded, and so is the actual progress of the task. Afterwards, a transcription is made of the session which can be analyzed in a variety of ways.

For my modeling tool evaluation method TAP sessions can of course be used as well. However, the method I propose differs somewhat from how you would do a survey. First of all, the participant would no longer directly control the scoring on most of the heuristics. This scoring is instead done by the researcher

based on what he observes in the participant, and on the thoughts uttered by the participant. First of all,a normal TAP session is organized. The participant should be asked to complete a certain task, preferably creating a model using the selected tool. Both the modeling process and the verbalizations should be recorded and time stamped. After the session is over, the researcher should create a transcript of the verbalizations. The transcript and the recording of the modeling process should then be analyzed and tagged according to each interaction pattern. So if we look at CaseTalk, the participant might be at time stamp X interacting with the system based on the *qualify part of expression* usecase. Then it is up to the researcher to distill from the transcript and the user's actions the scoring of the heuristics. So instead of having the participant score the heuristics, the researcher does it himself.

The main advantage of performing TAP sessions in this case is that it allows the researcher to make a judgement based on what he himself thinks. This means that all heuristics will be scored based on the same scale of good and bad which will lead to more uniform results. Also, the transcript may even reveal certain details that would otherwise be missed. Furthermore, a TAP session may be seen as being more scientifically sound then a survey [7], since it offers a score which has more evidence behind it.

However, there are two major downsides for choosing to evaluate with a TAP session. The first is fairly obvious: It is a lot more work. Without significant resources you will never be able to achieve the numbers of participants which may be reached using surveys. Think about it. Every single session needs to be transcripted and analyzed. That alone might take hours for every participant. A survey can simply be interpreted by a computer. Because of this reason, TAP sessions are rarely as big as surveys.

The second major downside of using TAP sessions is that the researcher himself will actually have to be familair with the modeling process and the tool he is evaluating. This fact leads to two new problems. First of all, during his familiarization with the tool, the researcher may already have formed an opinion on the tool which colors his interpretation of the TAP session's transcript and recording. Secondly, unless the researcher is actually an active modeler himself, he will never be as familair as the people he is observing which may lead to errors in the interpretation of what they are doing and what they are thinking.

For the case study in chapter 4 I will be doing a TAP session, so for more information and a worked out example for FCO-IM and CaseTalk I would like to refer you to that chapter. A survey would be advisable as well, but for this thesis I have decided to just test the method using a TAP session.

3.3.3 Comparing the Survey and TAP methods

To conclude this section I have created an overview of the advantages and disadvantages of my proposed methods of data gathering. Of course, this analysis is rather limited in it's scope, and I am quite clear that there is more to be said on this matter, but that is not the scope of this thesis. The overview may be found in table 3.4.

Research method	Simple Survey
Advantages	Easy to reach large population
	Easy to interpret
Disadvantages	Not a lot of data gathered

Resarch method	Advanced Survey
Advantages	Easy to interpret
	Depth is variable, more information
	Better statistics
Disadvantages	Getting large population more difficult
	More work for researcher

Research method	TAP Session
Advantages	Large amount of information
	Better evidence for conclusions
	May lead to new unexpected insights
Disadvantages	A *lot* of work is needed
	Participation must be lower because of work
	Researcher has to be familair with tool

Table 3.4: Overview of research methods

Chapter 4

Case Study Definition

In this chapter I will introduce and describe the case study I will perform to evaluate my evaluation method. I will introduce the various test groups which I will use in the case study, and specify the exact nature of the case study itself. My aim in this chapter is to provide a general overview of the case study, in the hope that it is repeatable for future work.

4.1 Case Study Goals

By performing a case study I hope to perform a field test of my evaluation method (as described in chapter 3). By performing a field test I hope to see if the method works in practice, and if the results yielded by one of the two evaluation methods discussed in section 3.2 are consistent with eachother. My case study will focus on testing the FCO-IM modeling tool CaseTalk.

The case study results will not only be viewed as a test of CaseTalk, but will also be analyzed to ascertain how good the evaluation method itself is. How this is done is explained in the relevant sections, further on in this chapter. To summarize:

- *Case Study Goals*

    1. Perform an evaluation of CaseTalk
    2. Evaluate the evaluation method itself

4.2 Case Study 1 - TAP Session

4.2.1 TAP session description

To evaluate my evaluation method, I will start by performing a TAP session on the FCO-IM modeling tool, CaseTalk. The TAP session will more or less be organized in the fashion described in section 3.3.2. This means that a task will be

assigned to the students which will be evaluated. In this case I will record two sources of information: Their verbal communication, using a microphone, and their interactions with CaseTalk, which will be recorded using a screen capture application called SnagIt (more information at http://www.techsmith.com/screen-capture.asp). These two sources of information will be carefully time stamped to allow for future analysis.

The investigator performing the study (in this case, myself) will also lead the experiment. This means that the investigator will need some basic knowledge of FCO-IM, in order to properly monitor the experiment as it takes place.The main task for the investigator is to observe, and make notes where applicable. It should be noted however, that the investigator is not allowed to comment on the experiment itself while it is taking place. The research subject should react and act as if no observer is present. This is to prevent the investigator having any influence on the outcome of the experiment. However, the actual presence of the researcher is still very important ([15]) since it is very well possible that relevant information for the experiment is not actually recorded in the process. Also, the investigator is also responsible to correct the participant is he does not adequately verbalize his thoughts during the assignment.

The main part of the TAP session will consist of the participant performing a modeling assignment (of which the details will be discussed later) following the rules of FCO-IM, and making use of CaseTalk to perform the actual modeling. By use of an instruction sheet and instructions by the investigator the participant will be given his instructions: Simply complete the modeling assignment as you would do on any other occasion, but be sure to verbalize all your thoughts with regards to performing the interactions with CaseTalk. A time limit of 10 minutes will be set on completing the assignment, to reduce the complexity of the results, and reduce the amount of work needed to process the results. As stated in various literature (including [15]): a small experiment can still yield results relevant for a larger whole, when properly set up and executed.

The role of the investigator in this part will be two-fold: First, he has to observe what is going on and encourage the participant to verbalize more when so needed. Second, the investigator has to monitor the actual modeling session and make sure the participant does not stray too far from the given goal. Of course, a little room for error should be kept since this could be a result from the modeling tool itself. However, when the participant gets stuck in completing the task, the investigator should assist in order to make sure the experiment keeps moving. Keep in mind that this is in essence not about the actual modeling: this only serves as a tool to achieve our real goal: Evaluate CaseTalk. This means that a little involvement from the investigator is justified.

To reduce influence and interruptions from external sources, the experiment will take place in a closed room where only the participant and the investigator will be present. Food and drinks are not allowed during the experiment, as it might disrupt the participant in completing his assignment. Besides, the short nature of the experiment negates any need for refreshments during its course. The participant will not be allowed to use his own computer system as we want all participants to use the same version and settings of CaseTalk. All other

equipment will also be prepared and tested before the experiment takes place to reduce any chance of unexpected delays resulting from malfunctioning or improperly set up equipment.

The timing and planning of this case study was done using the expertise of the staff of the Data Architectures & Metadata Management Group. Using their knowledge of the student's time table I have selected a date and time on which no classes were planned and which was not in advance of a test for which preparation was required. Also using their help, I have contacted all students through e-mail to ask for volunteers who would be willing to sacrifice some of their time to participate in my experiment.

Finally, two of the TAP sessions will be done with an experienced modeler to ascertain whether or not including such person has any effect on the outcome of the experiment. Currently, I have not yet researched this possibility, so the outcome of this little meta-experiment may lead to future research. My hypothesis is that there will not be a discernable effect based on the expertise of the modeler, but there might be an slight difference based on the background of the modeler. More research on this subject is needed as well.

4.2.2 Assigning the heuristics to use cases

As I have stated in section 3.3.2, use cases will need to be assigned to heuristics. This is so we can identify which interaction has to do with which heuristic, which in turn will allow me to perform a more to-the-point analysis of the TAP session's resulting transcripts. Keep in mind that I am looking at the interaction between the user and the modeling tool. These interactions are defined by use cases which I identified in section 2.3.2. Since I am looking at these interactions, and the heuristics are based on interactions between man and machine it is therefore a valid conclusion to say that every use case (or interaction) has one or more heuristics which have a bearing on that specific use case.

In order to determine which heuristic has a bearing on which use case one can take two approaches: assigning use cases to the heuristic or assign heuristics to the use cases. The difference between the two is not very important for the end result, but it does matter in actually getting to it. If we were to use the first approach, I would look at every heuristic, and then decide which use case is relevant for that heuristic. This means that I will be working through the list of use cases every time. The second approach will require me to look at every use case, and then decide which heuristic is relevant for this specific use case. By using this approach, I will be going through the list of heuristics every time. Since the list of heuristics is significantly shorter then the list of use cases, this will reduce some of the workload. This will be true not only for this specific TAP session, but for every TAP session you might undertake for other tools as well, since the list of use cases will be in most (if not all) cases be longer than the list of heuristics; unless you have a very limited tool at your disposal.

The combination of use cases and heuristics can be found in table 4.1. An in depth explanation of the various use cases will be given as well. Note that the naming of the use cases is the same as the one used in section 2.3.2, and

Use Case (name)	Relevant heuristics
(1) Enter expression	1,2,3,4,5,6,7,8
(2) Qualify part of expression	1,4,5,6,8
(3) Confirm expression is qualified/quantified	4,7,8
(4) Confirm expression	4,7,8
(5) Undo QlaQua step	2,4,7,8
(6) Cancel expression	2,3,4,6,7,8
(7) Delete repository item	1,2,3,4,6,7,8
(8) Generate new IGD	1,2,3,7
(9) Add repository item to IGD	1,2,3,4,5,6,7,8
(10) Delete repository item from IGD	1,2,3,4,5,6,7,8
(11) Add unicity constraint	1,2,3,4,5,6,7,8
(12) Delete unicity constraint	1,2,3,4,5,6,7,8
(13) Add totality constraint	1,2,3,4,5,6,7,8
(14) Delete totality constraint	1,2,3,4,5,6,7,8

Table 4.1: Use cases with assigned heuristics

the numbering for the heuistics is taken from section 3.2.1. This is list is not completely set in stone, however, as some instances of use cases may not be related to the heuristic they are generally related to. However, when such an event takes place I will identify it, and take this into account when completing the evaluation. Also, although for most use cases *all* heuristics are relevant, I have chosen to list only the ones in which there is an emphasis.

- *Enter expression.* This use case is particulary relevant for the modeling process as a whole, since it is one of the core aspects of entering data. Therefore, since the use case itself is very important to the modeling process, all heuristics are also important to consider for this specific use case.
- *Qualify part of expression.* Qualifying a part of the expression is very relevant for the modeling process. Therefore, heuristics 1, 4 and 8 are very important, since the qualification process needs to be the same as the theoretical qualification process done on paper, and also be without errors. Futhermore, the process of qualification needs to be done quickly and efficiently as well, so heuristics 5 and 6 are also important.
- *Confirm expression is qualified/quantified.* This use case is a confirming step, which means that it only functions to confirm the final part of a certain part of the ClaQla process. Because of this, error detection is still important so heuristics 4 and 8 need to be adhered to. Furthermore, we want the confirmation process to be clear, so heuristic 7 is also relevant here.
- *Confirm expression.* The same conditions as in the previous use case apply, so heuristics 4,7 and 8 are relevant here.

- *Undo ClaQua step.* The very name and description of this use case indicate that heuristic 2 is relevant. Furthermore, this use case is about error control, which means that heuristics 4 and 8 are relevent here too. Finally, this needs to be done efficiently, so heuristic 7 should be applied as well.

- *Cancel expression.* This use case too is about correcting errors. It is logical to assume that when you cancel an expression completely you were obviously making a mistake. Therefore, heuristics 4 and 8 are very important here. However, to prevent mistakes, this action should not be taken lightly, and the user needs to know exactly what he is doing when selecting this action. Because of this, heuristics 3, 6 and 7 should be adhered to. Finally, because of the serieus impact of this action, heuristic 2 applies as well, since the user should be able undo this action when he has made it.

- *Delete repository item.* More or less the same reasoning applies here as with the previous use case, but with the adition that heuristic 1 is relevant as well. This is because deleting a repository item is dependent on so many other items in your repository. If you delete a fact type for example, all related object types and expressions relating to that fact type will have to be deleted as well; or rather: *should* be deleted as well, depending on how good your tool is.

- *Generate new IGD.* Generating a new IGD is dependent on the items you have in your repository. Therefore, the tool should take care to identify its repository items correctly; hence the need for heuristic 1 to be applied. Furthermore, the action should be easy to use and quickly accessable, so heuristics 2,3 and 7 are relevant as well.

- *Add repository item to IGD.* Since this interaction is a crucial task in actually generating the IGD (and thus, the model itself) all heuristics must be applied. The interaction should be easy, straightforward and checked for any errors on the user's part.

- *Delete repository item to IGD.* Same reasoning as the previous use case, with the adition that heuristic 1 is even more important. When deleting an item the system should also delete all related items from the IGD, as you would otherwise have connections or constraints refering to nothing, so the system needs to know the rules of FCO-IM.

- *Add unicity constraint.* Again, this is a very important modeling operation which involves error checking, so the operation should be safe and quick. Therefore, all heuristics apply.

- *Delete unicity constraint.* Same as above. However, the system should also be aware that when deleting constraints, it should also re-verify the population. This is something that should be monitored by the investigator when conducting the experiment.

- *Add totality constraint.* Same as unicity constraint.

- *Delete totality constraint.* Same as unicity constraint.

To conclude this section I would once again like to emphasize that the above list only serves as a guideline. Since both the heuristics and the use cases are based on defintions in natural language they are not by any means to be considered as absolute. For this they would have to be formalized, which is not yet possible. Please use the above list only as a reference to what is *very* important for any given interaction and when you are conducting similair research, feel free to shift the focus to other areas. The main reason I have chosen to even make such a list is that it will allow me to conduct the TAP session in a more organized manner, and also to make analyzing the results a bit easier and systematic.

4.2.3 TAP Session assignment

Before I explain the assignment used in the TAP session, I would once again like to stress that the assignment has nothing to do with modeling itself. I will not even look at the quality of the model, or if the participants even successfully completed the assignment. What I am interested in, are their interactions with the system while trying to complete a modeling assignment. Because of this fact, the assignment is deliberately made very easy. Instead of making the assignment difficult, we looked at choosing an assignment which incorporated most of the FCO-IM elements requiring interaction with CaseTalk. I would also like to emphasize that the assignment was selected and specified by Dineke Romeijn, from her position as teacher. For this I am very grateful.

The assignment chosen is based on an assignment in [**?**] on page 28 and the assignment in [**?**] on page 18. We choose this assignment based on three distinct criteria which makes them successful for a TAP session (based on the criteria in [15]):

- It is short - can be completed in less then 15 minutes.
- It is simple - it requires very little skill to model the problem.
- It contains all elements of FCO-IM which in CaseTalk leads to interactions with the program.

The assignment itself was modified from the original version to be a little bit easier and so that it may be completed faster. The latter was done by removing the need to actually perform the first step of the modeling process (see figure 2.3 and the related section for more information) and thus provide a very clear set of factual sentences and constraints. This allows the assignment to focus on the part of the modeling process done in CaseTalk, namely the classification and qualification process, and the drawing of the IGD. The resulting assignment was checked and verified by Dineke Romeijn.

The model which the students are supposed to create can be seen in figure 4.1. However, instead of having the participants directly create that model, I have made it a little bit more complicated. First, the students will be provided with a population and a set of constraints and asked to create a model from

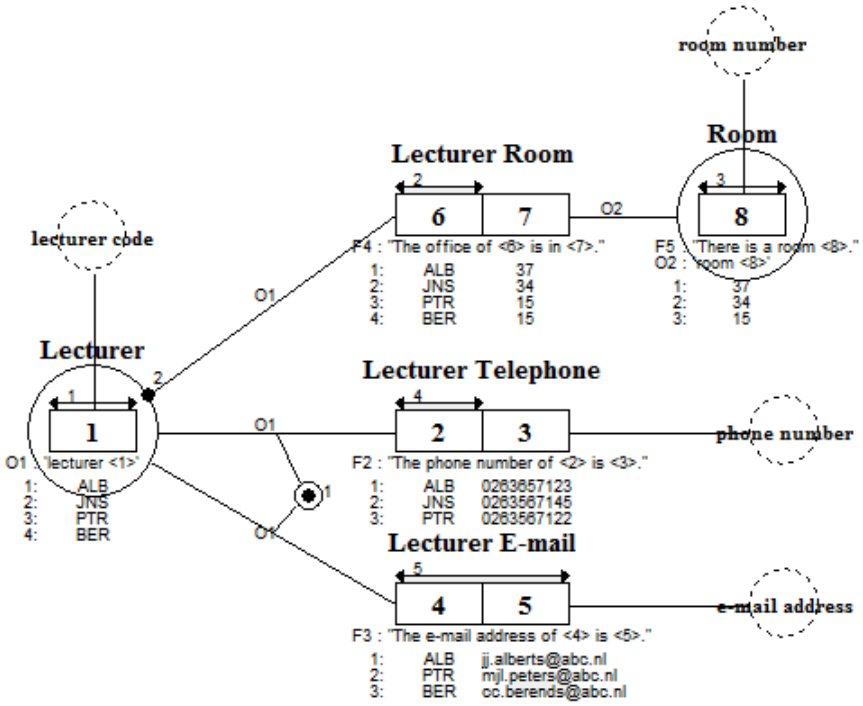


Figure 4.1: Resulting model for TAP session

that data. Using this task, I will observe to what extent heuristics 1, 2, 3, 5, 6 and 7 are adhered to.

However, the population given will have some critical points in which it is in violation of the constraints. The user is then asked to use any means necessary to correct those mistakes in the population and come up with a valid model. This step will allow me to observe to what extent heuristics 4 and 8 are adhered to. Because the user will have to use his imagination to correct the population (and come up with facts that can be used to complement the missing population elements) this latter part will allow for some deviation between the various outcomes, but it is my opinion that these deviations in end result will not affect the outcome of the experiment. This opinion is also stated in [15]: "*The end result of a certain experiment has little impact on the data collected by the experiment itself, as long as the participants have finished the same task.*".

Between the first and second part of the TAP session, the investigator will check the model produced by the modeler in order to make sure he has not strayed too far from the intended model. This is to try and achieve as much similarity between the various experiments as possible. If the resulting models are extremely different, we can no longer guarantee the external validity of the experiment. If the model is judged to be too different from the 'standard

model' seen in figure 4.1 then this standard model will be given to the student to work with.The investigator will however not be involved with the second part of the assignment. Finding the errors is the actual task to complete, so if the user cannot find the errors (either by reasoning or by use of CaseTalk's functionality) then he will simply be allowed to fail.

The first population will be given as fact sentences. These should easily be translated into expressions used by CaseTalk by the participants, since they can almost be translated without change. The only thing the participants are still required to do is to identify the labels, facts and objects; but since these are fairly obvious from the fact sentences I do not expect to see any big differences between the various models. The given fact sentences look as follows:

- The office of lecturer ALB is in room 37
- The office of lecturer JNS is in room 34
- The office of lecturer PTR is in room 15
- The office of lecturer BER is in room 34
- There is a room 37
- There is a room 34
- There is a room 15
- The phone number of lecturer ALB is 0263657123
- The phone number of lecturer PTR is 0263657145
- The e-mail address of lecturer ALB is jj.alberts@abc.nl
- The e-mail address of lecturer PTR is mjl.peters@abc.nl

The constraints which should be put in place on the model are given as sentences in natural language. I have decided not to simply state them as constraints, because I want to see how people translate the constraints into constraints which can be understood by CaseTalk. It will be made clear to the participant that all constraints are correct, and that all errors identified are the result of the population being wrong. Therefore, I can assume that all the models will have the same constraint unless an error was made. The constraint sentences given to the participants look as follows:

- All lecturers must have an office to work in.
- All lecturers should have either a telephone number, an e-mail address, or both.
- A lecturer can only work in one office. Offices may be shared by lecturers.
- All offices can only have one room number.

- A lecturer can only have one telephone number.
- Telephone numbers may be shared by lecturers.
- A lecturer can only have one e-mail address, and e-mail addresses can only be given to one lecturer.
- A lecturer is identified by his code, which can only be given to one lecturer.

The main problem that should be identified by the participants is the fact that the second constraint is violated. There are two teachers who do not have either a telephone number or an e-mail address. These errors are supposed to be corrected by the participant using either common reasoning (which should be verbalized) or by using any of the checking tools available in CaseTalk.

To summarize the assignment, I have listed and summarized the two distinct steps in the assignment:

1. Use CaseTalk to create an FCO-IM model based on the facts given. Do not deviate from these facts, all errors you may identify are intentional.
2. Use any means necessary to identify the errors in your previous model. Do not change any constraints, but adapt the population in such a way that all constraints are adhered to. Make up facts which are not currently given; as long as you make the population fit the constraints.

The time limit given for the assignment is currently set on 15 minutes. These 15 minutes are a guideline however, and it is up to the investigator's judgment to ascertain how long a session is allowed to take. If the participant is no longer making any progress at all, and the 15 minutes have elapsed it is logical to stop the experiment. If the participant is hard at work at around 15 minutes, it would be advisable to continue the experiment and let the participant finish his current task. Since I will be playing the investigator role in this specific experiment, I will list the criteria used to stop the experiment at the experiment's description listed in chapter 5.

4.3 Case Study Population

In this section I will discuss the intended case study population. Please note that there is a difference between the intended population and the actual population which participated in the experiment. For more information on the actual participation I refer you to chapter 5.

In order to perform a good case study, the chosen population has to be representative of the chosen domain ([15] and [7]). As easy as this may sound, this is actually one of the hardest parts of setting up a case study. If we look at the domain of CaseTalk, we can see that it is used by various groups of which professional modelers, educational staff, researchers and students are probably the most numerous. To perform a complete evaluation of CaseTalk (in order to achieve case study goal 1) I would certainly have to get a population containing

at least all of those people. However, since my resources are limited I have decided not to include researchers and professional modelers for the TAP session. This is mainly due to the fact that I can not get people from those two groups to participate at my current location. In order to take those groups into account I would have to spent considerable time traveling for only a ten minute TAP session, which is currently not feasible.

For the TAP session I have decided to have eight sessions. This number is based on the amount of work it will take to adequately process the results (as can be read in [15]). Because I will have to do everything myself, the amount of work should not exceed reasonable limits with regards to the size and time taken for this thesis. Six TAP sessions will be done with students, and two will be done with educational staff. The students will be selected from the HAN university by way of invitations. The educational staff will also be selected from the HAN university, and also by way of invitations. Through this process I can reduce overhead to a minimum, and make sure that I get a diverse group. The students will be invited from two distinct programs. Three students will be invited from the part time bachelor education. These students are currently in their 'developing an information system' semester, and are following the course 'database design' by Dineke Romeijn. This course teaches them the basics of FCO-IM and the use of CaseTalk. These students are from the Netherlands, and may not be affluent in English. Therefore, their TAP session will be held in Dutch.

Three more students will be selected from the Data Architectures & Metadata Management Group's Master course. These students are mostly of a foreign (i.e. not Dutch) nationality and are studying at the HAN university. They are following various courses regarding modeling in general, FCO-IM and CaseTalk. Their main language is English, so their TAP sessions will be held in that language.

I have considered also getting some students from the Radboud University to participate, to get a group of students without any real experience with CaseTalk. This in order to get a fresh perspective from people who have not yet worked with CaseTalk. However, I rejected this idea for a number of reasons. First of all, the lack of experience might also mean a lack of opinion. It is very hard for people to form an opinion on something on which they have only worked with for a few minutes. Even looking from the investigator, it is very hard to actually judge the quality of a tool when the people working with the tool have had no previous experience with it.

Finally, the two educational staff members are Dineke Romeijn and Chris Scholten, who both work at the Data Architectures & Metadata Management Group. They have extensive experience both in working with CaseTalk, and in teaching it to students. However, they have not had any significant influence in the development of either FCO-IM or CaseTalk, which makes them less biased. Guido Bakema and Jan-Pieter Zwart were also considered, but since they have basically developed both FCO-IM and had a significant influence on the development of CaseTalk I considered them to be too biased to provide objective information.

Chapter 5

Case-Study Results

In this chapter I will discuss (in depth) the results from the TAP sessios I have performed. I will discuss each experiment individually, and perform the evaluation method on each experiment. The conclusions drawn from these experiments can be found in chapter 6; this chapter will only provide the data and results themselves. All transcripts can be made available as attachements to this thesis, but will not be included in the thesis itself. Also please note that some of the experiments were performed in Dutch. English translations can be made available on demand.

5.1 General impressions

As the TAP sessions performed for this thesis were not just to actually get a score for CaseTalk and FCO-IM (but rather to investigate if my proposed evaluation method can return a good and consistent result) it is worth to mention a few observations made during the experiment. These observations might help to improve the method in future work, as it turned out that some of my assumptions had been proven faulty during the course of the experiment itself. These errors will be listed and explained.

First of all, I underestimated the amount of skill with the modeling tool required to perform the experiment. In section 4.3 I argued that skill with the modeling tool is not a very high prerequisite since even observing someone trying to work with the tool would be worth investigating. However, during the course of my experiments I encountered several participants who had in fact a somewhat low skill level with CaseTalk, and these experiments did not go very well. The participant would be unable to even perform some of the tasks set out in the assignment, leading to the person just randomly clicking around in CaseTalk and not knowing what to do. Since the participant was therefore not actually performing any of the use cases laid out in section 2.3.2, I was unable to gather any valid data from these actions leading to a unsatisfactory experiment.

The second mistake I made was to not asses my case study population care-

fully enough. When selecting participants I assumed a certain level of skill with the modeling method (FCO-IM) itself based on the fact that they were students or staff of the HAN and should be working with FCO-IM during their work or studies. This turned out to be a faulty assumption, since especially some of the students who participated showed a very low level of skill in working with FCO-IM. When this is the case, the entire experiment should be rejected outright, since such a person would not be able to demonstrate the connection between a modeling tool and a modeling language. If the person does not know what he is doing with the tool, then we cannot properly judge the quality of the heuristics defined in section 4.2.2 since he or she will have no clear goal in the tool; instead he or she will just be clicking at items in a random fashion to try and figure out what to do. Of course, this could be interpreted as valid data which says something about how well a tool can help you to learn a certain modeling language, but as I have not defined my heuristics in this way, I decided to reject the data gathered from these participants.

Fortunately, not all my general observations are negative. First of all, the assignment itself proved to be very effective. During the experiment I observed that the setup of the experiment required the participants to use most of the use cases described in section 2.3.2, which means that scores can be given for all of the heuristics. Furthermore, the assignment proved to be more or less as difficult as I expected it would be (for the population which had a certain level of skill in both FCO-IM and CaseTalk), and therefore the lenghth of the experiment proved to be within bounds as well. I needed to hurry the participant along during certain experiments, but this was mostly due to the perfectionist nature of the participant in question, and not because the assignment proved to be too much.

Something I did not consider in advance (but still proved important) is the fact that the domain of the assignment determines where the difficulty lays. In my assignment for example, the domain was of a educational nature with teachers being assigned to rooms and such, as can be read in section 4.2.3. This means that all of the paricipants, being selected from an educational institution, had the domain knowledge required to interpret the results. When performing this assignment in some other settings, care should be taken that the domain used in the assignment is familair to the participants as not to confuse them with terms they are not familair with. Again I want to emphasize that the main purpose of these assignments is to perform my evaluation method, not the check wether or not a participant has the ability to interpret domains and model them.

One further general observation is that the location I chose for some of the experiments was not ideal. My first location was a meeting room with open windows to a busy hallway. This meant a lot of distraction both in terms of the participant being distracted, and the audio recording being polluted with noise from the hallway. The second location was a classroom which could be locked and closed, and had no windows facing inwards. I found that the participants in this room were more able to focus their attention and that the audio recording was much clearer.

5.2 Experiment results

The results from my experiment will be described in detail for each experiment in turn. Every experiment will be described (in terms of setup and participants) and I will discuss my observations from every experiment. The details and transcripts from the experiment can be found as an additional document and will not be included in this thesis. Finally, I will perform the evaluation method described in chaper 3 to determine a score for CaseTalk based on the participant's actions. The final score for CaseTalk will be given in the next section.

Experiment 1

Experiment introduction

The first experiment was done with a master student from the HAN university (information published with permission of participant). She is studying FCO-IM in several courses and has performed small assignments with FCO-IM before. Her knowledge therefore is mostly theoretical, but she was convinced she could perform the small assignment put before her. The experiment itself took place in a closed meeting room where we were unfortunately disturbed due to the poor quality of sound isolation present in that room. The audio recorded is likewise polluted with random background noise, making it harder for me to properly finish transcripting this experiment.

Experiment overview

The participant started out by entering all the expressions given before her. This meant that she was doing a lot of redundant work since most of the sentences describing population are very much the same. Use cases 1, 2, 3 and 4 were used to enter the expressions. While doing this work the participant observed that *'I don't really need to give it a name since I already have the facts which is about the same (02:20)'*, while manually trying to find the appropreate fact type already stored in CaseTalk. By doing this, she was in essence repeating steps she already took, and could furthermore make mistakes had she not recognized this. Because of this, heuristics 4, 5 and 6 were violated.

Later on in the experiment, the participant said: *'this was just a population, I didn't need to create a new expression for it since I already have an expression (03:13)'*, and proceeded by cancelling her work only after she had in fact already finished it. By doing this, she demonstrated that CaseTalk had in fact not informed her about the fact that she had already entered a completely similair expression and was in fact only adding population to her model. Time was wasted because CaseTalk did not properly keep her informed about what she was doing during the execution of use cases 1, 2 and 3. Again, heuristics 4, 5 and 6 were violated because of this.

6 minutes into the experiment, the participant was trying to define her existency postulating fact types about the object type 'room'. However, she did not really know how to perform this action in CaseTalk, and ended up creating

double object types at (06:10). CaseTalk should have (based on the fact that the modeler should reflect the modeling language) have given her the choice wether or not she wanted to create a new object type, or to define an existency postulating fact type about the object type 'room'.

The participant then proceeds with entering all the fact type expressions (and population) in a similair fashion using the procedures defined in use cases 1, 2, 3 and 4. Only after 10 minutes does she (by accident) discover the fact that CaseTalk allows the editing of populations using a database editor (which is in fact not related to the way FCO-IM works). So to summarize, CaseTalk is violating the way popultion is added in FCO-IM (heuristics 1 and 3) by implementing another way of doing this. Furthermore, this method is flawed in that it is apperrantly not obvious for this user how to work and find it.

For the next phase of the assignments, the participant creates an IGD as you would do in FCO-IM. However, the tool does make her job a bit harder, since the IGD she creates by dragging all her repository items into the diagram at (10:29) becomes an instant mess with overlapping lines, boxes and circles. This happened by executing use cases 8 and 9 and is in violation of heuristic 5. She corrects this by manually putting all items in a more logical position.

Adding the constraints in this experiment went very well. By executing use cases 11, 12, 13 and 14 the participant was able to quickly put the relevant constraints onto the model in a correct fashion. Because of this, I can conclude that for these use cases actually all heuristics were followed. For this participant it was easy to do and she seemed to be guided by the system when placing constraints. Furthermore, the system would only allow constraints which are legal in FCO-IM.

Experiment conclusion

The scoring for experiment 1 has been decided as follows: (based on the description given of the experiment, and the general remarks about the heuristics made in this description)

Heuristic Name	Score
Match between system and the real world	+
User control and freedom	o
Consistency and standards	-
Error prevention	- -
Recognition rather than recall	o
Flexibility and efficiency of use	o
Aesthetic and minimalist design	-
Help users recognize, diagnose, and recover from errors	o

Table 5.1: Scoring Experiment 1

This leads to the following score for CaseTalk, based on the calculations demonstrated in chapter 3.

Absolute score	Adjusted score	Final score
-3	-5	2.92

Table 5.2: CaseTalk score Experiment 1

As we can see, the experiment did not go well for CaseTalk. This was mostly due to the problems with preventing errors and providing one good overview of how far along the modeling process was (heuristics 1 and 4). Furthermore, based on this experiment we did not really see any strong points of CaseTalk except for the match between the constraint process in CaseTalk and that same process in FCO-IM.

Experiment 2

This experiment was rejected after analysis indicated that the participant's level of skill in both FCO-IM and CaseTalk were below the requirements to successfully complete the assignment. A recording of this experiment is available, but I do not consider that data in this recording to be valid based on the general observations I made, which are described in section 5.1.

Experiment 3

Experiment introduction

The third experiment was done with a night course bachelor student at the HAN university (information published with permission of the participant). He has recently done a course on FCO-IM and has worked with CaseTalk during this course. This person, however, is a bit skeptical about the use of FCO-IM. Since he is a database manager, he wants a model to be translatable to a database structure. FCO-IM however does not yet support a real time connection between an actual database and the model, something that according to this participant makes FCO-IM useless for any real-world application. He also does not have a very high opinion of CaseTalk, which he considers to be flawed and riddled with bugs. Nevertheless, he agreed to approach the assignment with a neutral view. This experiment took place in the same location as the first experiment, which lead to the same problems and complications in terms of the sound recording.

On a final note, this experiment took place in Dutch. Translations of all citations will be provided in this description.

Experiment overview

This time the participant also started with entering all of the expressions provided in the assignment. As with participant 1, he also did this step by step; entering each expression individually, as opposed to making use of the population editor provided by CaseTalk. While doing this, the participant remarks: *'nou, ik heb nog iets niet goed gedaan, maar ik ga maar even door (02:20)'*, or: *'well, I have still made a mistake, but I will just continue'*. In this case,

his mistake was a simple spelling error made in the word 'lecturer' which the participant types as 'lecture'. Because of this, CaseTalk did not recognize this expression as being part of an already existing fact type, and tried to make it a different one with the same name, leading to a 'recursive structure'. The participant did not (at first) realize what he did wrong, and did also not understand the error message CaseTalk displayed to him; something which is in violation of heuristics 4 and 8, during the excecution of use cases 1, 2 and 3. The participant never realized his mistake; instead he chose to enter the expression again thereby inadvertently correcting his previous mistake.

During time index 04:20, the participant was starting on his existence postulating fact types. As we already saw during the first experiment, this lead to some confusion. The mistake made by the participant in this instance was that he tried to connect a fact type to an already established label type which lead to an error from CaseTalk during the excecution of use case 3 as it should in FCO-IM. This means that in this case heuristic 1 has been clearly followed. However, the error itself was simply a message saying that he had performed an illegal function without further explanation. This lead the participant to say *'ik, uh... ik moet even wat he? ik geef het even een andere naam (04:59)'*, or *'I, uh... I have to do something, right? I'll just give it another name for now'*. He did not know what to do (and CaseTalk gave him no advice) so he simply tried something else, which lead to a domain error in the resulting model, violating heuristics 1, 2 and 3.

When trying to generate the IGD for the next phase (adding constraints) the participant could not remember how to actually create one. Using tooltips and clicking around did not help him either, so I had to assist him in this task. This seems to be a violation of heuristics 6 and 7 while executing use case 8.

After eventually generating the IGD he also had the trouble of actually creating a layout that would make it easily readable to a viewer, as was the case in experiment 1. Adding constraints turned to be impossible for this participant. His skill with FCO-IM and CaseTalk proved to be insufficient to actually intepret the fact sentences and translate those into constraints. Therefore, it is impossible to judge any of the use cases and their associated heuristics based on this experiment.

Finally, the participant used the build-in rule checker to determine if there were any errors in his model. The participant did not understand any of the information given in the error display, after which he stated: *'verify well-formedness, dan gaat hij dus de regeltjes checken en dan roept hij dus vanalles wat niet goed zit'* or *'verify well-formedness, and he will check the rules and will tell me everything that is not right at the moment'*. So apperrantly the information in that display was formulated according to FCO-IM standards, and with enough expertise he would have been able to identify one or two errors in his model. If we consider the heuristics for a moment, I can conclude that heuristics 1 and 8 have been adhered to.

Experiment conclusion

The scoring for experiment 3 has been decided as follows: (based on the description given of the experiment, and the general remarks about the heuristics made in this description)

Heuristic Name	Score
Match between system and the real world	+ +
User control and freedom	o
Consistency and standards	o
Error prevention	- -
Recognition rather than recall	o
Flexibility and efficiency of use	-
Aesthetic and minimalist design	-
Help users recognize, diagnose, and recover from errors	+

Table 5.3: Scoring Experiment 3

This leads to the following score for CaseTalk, based on the calculations demonstrated in chapter 3.

Absolute score	Adjusted score	Final score
-1	-1	4.58

Table 5.4: CaseTalk score Experiment 3

As we can see, the experiment did not go well for CaseTalk. First of all, the link between FCO-IM and CaseTalk is (according to this experiment) very good. However, error prevention has only been shown once in this experiment, and it did not help at all. Furthermore, it allowed a lot of errors to take place which might have been prevented. Finally, the error checker showed me that CaseTalk does offer some support in actually finding errors in the model, but you do need a strong understanding of FCO-IM to be able to work with this tool

Experiment 4

Experiment introduction

The fourth experiment was also done withe a night course bachelor student at the HAN university (information published with permission of participant). He too has done a course on FCO-IM and CaseTalk. In contrast to his fellow student from experiment 3, he has a higher opinion of FCO-IM but he is also skeptical about CaseTalk; considering it too buggy. He also agreed to approach the assignment from a neutral standpoint. This experiment took place in the same location as the first and third experiment, which lead to the same problems and complications in terms of the sound recording.

On a final note, this experiment took place in Dutch. Translations of all citations will be provided in this description.

Experiment overview

This participant also started with entering the first expression of the assignment. The first few went really rapidly with the participant being able to connect his knowledge of FCO-IM with the CaseTalk interface. However, the lack of experience for this participant led him to make a rather crucial mistake: He tried to identify part of the expression with the expression itself, as seen on time stamp 01:43. The message given by CaseTalk did not allow him to find out where the actual error in his expression lay, so the participant chose to turn the object into a label type, thereby circumventing the error and proceeding with executing use cases 2 and 3. Although the user did not know the error, CaseTalk did prevent him from making a crucial mistake, thereby fulfilling heuristic 4.

Following this, the participant decided to *'ik pak alleen de tweede nog want één tot en met vier lijken zoveel op elkaar dat ik denk dat als ik alleen... één tot en met vier neemdat dat wel voldoende is'*, or *'I will only take the second one, because one to four are so much alike that I think that if I only... take one until four it should be sufficient.* This is an interesting deviation from the path taken by the previous participants as this participant has decided to let CaseTalk do the work for him by simply identifying populations from expressions. In this, he has demonstrated that use cases 1, 2, 3 and 4 are more or less open to his own interpretation, thereby providing an excellent example of how heuristics 2, 4 and 6 are herein followed. This is confirmed when at 03:00 the participant states: *'en inderdaad, hij herkent mijn zin... match'*, or *'and indeed, it recognizes my sentence... match'*, and the CaseTalk has indeed already puzzled together the sentence for him.

The participant starts to gets frustrated when he has reached the existence postulating fact types. First of all, he explains that: *'alhoewel ik weet dat CaseTalk als je eenmaal een fout gemaakt heb dat het vreselijk lastig te repareren is'*, or *'although I know that in CaseTalk once you make a mistake it is very hard to repair it'*, a statement which is then proven in his actions. He makes the mistake of (again) recursively defining an expression. This leads him to continiously have to retype the expression since he could not make CaseTalk simply forget his last qualificiation. For use cases 2, 3, 5 and 6 this means that heuristics 2 and 6 were not followed. Eventually the participant got so frustrated on trying to define the existance postulating fact types that I decided to move him on to the next part of the assignment.

While modeling the IGD, this participant decided to drag all his elements to the canvas one-by-one, thereby creating the model as he saw fit. This did not only help him to create a readable model, but also proved that for use cases 8 and 9 heuristics 1 and 6 are fulfilled. The resulting model looked clean and tidy.

When adding constraints, things got complicated again. The participant did not remember how to add certain constrains. He knew what button to press,

but he was unable to select the appropreate amount of facts to actually put the constraint on. This is due to the fact that the interface forces the user to make a selection on how he wants to select one or more facts from the IGD. For use cases 11 and 13 this would seem to indicate that although heuristic 1 is followed (constraints are displayed exactly the same as in the real world), heuristics 2 and 6 are violated since the user is unable to actually add them to his model.

Experiment conclusion

The scoring for experiment 3 has been decided as follows: (based on the description given of the experiment, and the general remarks about the heuristics made in this description)

Heuristic Name	Score
Match between system and the real world	+ +
User control and freedom	+
Consistency and standards	o
Error prevention	o
Recognition rather than recall	o
Flexibility and efficiency of use	-
Aesthetic and minimalist design	-
Help users recognize, diagnose, and recover from errors	-

Table 5.5: Scoring Experiment 4

This leads to the following score for CaseTalk, based on the calculations demonstrated in chapter 3.

Absolute score	Adjusted score	Final score
0	3	6.25

Table 5.6: CaseTalk score Experiment 3

This experiment went a bit better for CaseTalk. The user used a different approach then seen before in the earlier experiments, which lead him to use some of the features of CaseTalk not seen before. The match between the system and the real world is still strongly shown, but for the first time error prevention is done correctly as well (the recursion example). Error prevention could still be a lot better though, as the participant's problems with the existance postulating fact types demonstrated.

Experiment 5

Experiment introduction

The fifth experiment was done with an employee and teacher from the HAN university. He teaches several courses on FCO-IM and modeling in general, but

does not have a lot of experience with CaseTalk (information published with permission of participant). He is enthusiastic about the ideas expressed in FCO-IM, but is also skeptical about CaseTalk for the same reasons given by the other participants. Because of my negative experiences with the first experiments, this experiment was held in a closed classroom at the HAN university, where I could not be disturbed by people walking by. This led to a much clearer audio recording, and greater concentration for the participant.

On a final note, this experiment took place in Dutch. Translations of all citations will be provided in this description.

Experiment overview

The participant started with entering the first expression which went well, as we have seen before. By classifying and qualifying (use cases 2 and 3) he managed to quickly create his first fact- and object typehis experiment went a bit better for CaseTalk. The user used a different approach then seen before in the earlier experiments, which lead him to use some of the features of CaseTalk not seen before. The match between the system and the real world is still strongly shown, but for the first time error prevention is done correctly as well (the recursion example). Error prevention could still be a lot better though, as the participant's problems with the existance postulating fact types demonstrated. . However, after this the participant decided not to manually input the remaining population (concerned with the same expression) but to leave this for a later date, arguing that he only needed the first part of the population to actually create the model. This is a good example of how heuristics 2 and 6 are followed; since CaseTalk allows the user to perform the tasks as he or she sees fit.

While entering the existence postulating fact type, the participant seems to know what he is doing, as opposed to the previous experiments in the participants had trouble entering these fact types. It should be noticed however, that although the participant knew what he was doing, it did not seem as intuitive as it is on paper. The user has to select an already existing fact type which under normal circumstances means you are adding an instance to this fact type. In this case, without warning, this function is suddenly changed meaning that heuristic 3 is being violated.

Another thing I noticed in the recording, is that when the participant says: *'ik klik elke keer te veel... (02:51)'*, or *'every time I click too often'*, he does seem to be clicking in thin air, apparently because he is suspecting another action to take place (most likely in use case 4). Unfortunately, I cannot confirm this so I cannot take this information into account.

At (03:40) the participant makes use of CaseTalk's built in function to identify fact types which have already been entered and automatically fill these in. I have already observed in experiment 2 how this can be a bad thing (since the smallest spelling error will already disrupt the process), but during this session the tool worked quite well, taking a lot of work away from the user during use cases 2, 3 and 4. Because of this, I can conclude based on this experiment that in these use cases heuristics 2 and 6 are nicely followed.

When the participant tries to generate the IGD, he runs into trouble again (use case 8). This is because he does not remember how to do it, and CaseTalk's interface does not offer any help. This can be seen as a violation of heuristic 7. Because the participant was losing a lot of time on this I assisted him by pointing him to the right menu item to create the IGD.

After creating the IGD, the participant adds all the repository items to the canvas at once (use case 9) which results in the randomly positioned items I observed earlier in experiment 1. The participant had to take some time to rearrange the items so that the diagram became readable leading again to a loss of time. This could be considered bad, since of the main goals of a modeling tool is to increase efficiency. To summarize: heuristic 6 is violated in this case, since time is lost.

Finally, this participant had a very big problem with adding constraints. Single-role constraints were no problem, since the recognized the icons from the FCO-IM standard (use cases 11 and 13, applied to heuristic 1). Selecting multiple roles however proved to be impossible. He simply could not manage to use CaseTalk's interface to select multiple roles. I observed that the buttons and icons in CaseTalk do not match their tool-tips, which in turn do not really describe the exact function anyway. This leads me to the conclusion that for use cases 11 and 13 heuristics 5, 6 and 7 are violated. The participant did eventually not figure it out in the allotted time.

Experiment conclusion

The scoring for experiment 5 has been decided as follows: (based on the description given of the experiment, and the general remarks about the heuristics made in this description)

Heuristic Name	Score
Match between system and the real world	+
User control and freedom	+ +
Consistency and standards	o
Error prevention	o
Recognition rather than recall	-
Flexibility and efficiency of use	- -
Aesthetic and minimalist design	- -
Help users recognize, diagnose, and recover from errors	o

Table 5.7: Scoring Experiment 5

This leads to the following score for CaseTalk, based on the calculations demonstrated in chapter 3.

Absolute score	Adjusted score	Final score
-2	1	5.42

Table 5.8: CaseTalk score Experiment 5

From this experiment I concluded that CaseTalk allows for a wide variety of approaches to complete your modeling work, and that those approaches match the real world implementation of FCO-IM on paper, which explains that good scores for heuristics 1 and 2. However, the 'secondary' heuristics all get bad to very bad scores based on what I have observed in this experiment. In this experiment, I especially observed the design and efficiency of CaseTalk to be rather poor in some cases.

Experiment 6

Experiment introduction

The sixth experiment was done with an employee and teacher from the HAN university. She teachers several courses on FCO-IM, and organizes and grades assignments made with CaseTalk (information published with permission of participant). She has extensive knowledge on both FCO-IM and CaseTalk and a lot of experience on modeling with CaseTalk as well. Her opinion on CaseTalk is rather mixed. She thinks the tool is well designed, but does not like the current version of the tool since it contains too many bugs. This experiment was also conducted in a closed classroom, leading to the same advantages as described in experiment 5.

Since this experiment was conducted with the most experienced participant, it was also the most complete. However, I will only look at the parts which were more or less completed by other participants as well, otherwise the final result would be unbalanced and statistically flawed. Furthermore, since this person teaches in CaseTalk, she seemed to be somwhat biased towards the tool and does not openly want to give it a bad name. I have tried to be observant of this fact.

On a final note, this experiment took place in Dutch. Translations of all citations will be provided in this description.

Experiment overview

The participant started by entering the first expression into CaseTalk and classifying/qualifying it, thereby executing use cases 1, 2, 3 and 4. This was in complete agreement with the standard approach in FCO-IM, so I can conclude again that for this part heuristic 1 has been nicely followed. She then also chose not to enter the population as expressions, but rather to enter the population at a later time using the population editor. Heuristics 2 and 6 are therefore followed.

Moving on to the existence postulating fact types, I could clearly see that the participant knew what she was doing. However, as I had already concluded

during experiment 5, the way she does it does not imply proper design. The action she performs would in another context imply a completely different course of action which is not explained by the tool itself. Therefore, people with less experience than this participant could be confused, something which is in violation of heuristic 3.

Because of her experience, the participant very rapidly completed the first part of the assignment, without allowing me to observe any very good or very bad elements of CaseTalk. The only think that is very clear however, is that for example at time indexes (02:50), (04:20) and (05:00) she demonstrates that CaseTalk can be used in such a way that the normal use cases are no longer followed, but a much more efficient result can nevertheless be achieved which is a good example of heuristic 2 being followed.

After creating the IGD, she decided to take the repository items one-by-one, and only take the items she deemed necessary. This is basically a variation on use cases 9 and 10 and an example of how heuristic 2 is again being followed. Furthermore, the easy drag and drop system is an example of how heuristic 6 and 7 are implemented as well.

Adding constraints was very easy for the participant, but while executing use cases 11 and 13 I could not help to observe that the interactions she had to take were rather vague. A lot of times she had to use a complicated combination of buttons (such as on time index (05:40)) to achieve a certain selection of roles. This is certainly not a good example of how heuristic 6 should be followed.

As a final word, the error checking capabilities have not been observed at all during the experiment due to the unusually high level of experience demonstrated by this participant, so they will be graded as neutral in the conclusion.

Experiment conclusion

The scoring for experiment 5 has been decided as follows: (based on the description given of the experiment, and the general remarks about the heuristics made in this description)

Heuristic Name	Score
Match between system and the real world	+
User control and freedom	+ +
Consistency and standards	-
Error prevention	o
Recognition rather than recall	o
Flexibility and efficiency of use	o
Aesthetic and minimalist design	o
Help users recognize, diagnose, and recover from errors	o

Table 5.9: Scoring Experiment 6

This leads to the following score for CaseTalk, based on the calculations demonstrated in chapter 3.

Absolute score	Adjusted score	Final score
2	5	7.08

Table 5.10: CaseTalk score Experiment 6

Based on this experiment, CaseTalk should receive a high grade. The important things (such as match with the real world, and flexibility) are followed, whereas the less important things are more or less neutral. I am speculating that the high score is related to the amount of experience this person has, as the experiments with the less experienced persons yielded lower grades. This will be elaborated upon in chapter 6.

Experiment 7

This experiment was rejected, same situation and grounds as experiment 2.

Experiment 8

This experiment was rejected, same situations and grounds as experiment 2.

5.3 Final score FCO-IM

To calculate the final grade for FCO-IM, we simply take all the grades from the experiments and take the average. The numerical results may be found in table 5.11. If we assume that me evaluation method is valid, we can conclude the following things about CaseTalk:

- CaseTalk did not score very well in the experiments, which was mostly due to the error prevention and correction being bad
- The match with the real world was usually very good
- With proper experience, CaseTalk can be used very flexibly
- Some parts of the CaseTalk interactions are not consistent, with the same action producing a different result based on unclear circumstances

These conclusions can of course be elaborated upon, but I would like to point out that the main point of this thesis is not to actually evaluate CaseTalk, but rather to evaluate my evaluation of CaseTalk. This will be done in chapter 6, so for more in-depth conclusions about the method itself please consult this chapter.

Experiment	Grade
1	2.92
2	N/A
3	4.58
4	6.25
5	5.42
6	7.08
7	N/A
8	N/A
Average	5.25

Table 5.11: Grades for CaseTalk

Chapter 6

Conclusions & Future work

In this chapter I will discuss the conclusions drawn from this thesis. This will mainly be about the testing of the evaluation method described in chapter 3 and what I could conclude from these experiments. Please note that this chapter will not be about CaseTalk or FCO-IM. For more conclusions on FCO-IM itself I would like to refer you to chapter 2. Conclusions about CaseTalk given the experiment results may be found in chapter 5.

6.1 General conclusion

When developing the evaluation method I discussed all the relevant literature and method on which I have based my method. Because of this, I can conclude that the internal validity of the evaluation method is therefore proven (given the fact that there are no errors in my research). What is much more interesting are the conclusions that I managed to get from the experiments as described in chapter 5.

I have already stated earlier that for the method to be externally valid, it should yield more or less the same result for every participant during the TAP sessions, if the researcher will objectively look at the way the participants work. However, as I observed during the experiments, it would appear that radically different scores can appear when the experience of the participants changes. As seen in experiments 3 and 4; participants with more or less the same amount of skill yield similar results. However, as is evident from experiment 6; people with a lot of experience with the tool will be able to mask the tool's shortcomings with their own personal experience and the way they work with the tool.

Based on this, a couple of changes can be proposed to my initial draft of the evaluation method. Selection of TAP participants should be much more intensively. 2 groups are probably needed: One group with little experience with the tool, and another group with a lot of experience with the tool. The results from these two groups could then be averaged to come up with a valid grade for the tool you are investigating.

Another change to the method would be that the TAP assignment should be build in such a way that ALL use cases will be used during the experiment. During my TAP session, some use cases were not used by any of the participants. This means that I could not really provide an evaluation for those use cases. The most common of these are the 'delete' and 'undo' variants of the use cases. If the participant does not make a mistake, he or she will have no reason to execute these use cases. Perhaps such 'mistakes' could be introduced in the assignment.

Another conclusion I managed to draw was that there is not enough information in my first draft about how the use cases in a program may be linked to the heuristics. For this, a set of guidelines should be created. These guidelines should give suggestions on what aspects of a use case define what heuristics are important for this use case.

6.2 Answering the research questions

1. How can we analyze a modeling tool from an HCI perspective?

   In this thesis I have developed an evaluation method based both use cases and usability heuristics from the HCI field. The main idea is to first identify the interaction patterns of a given modeling tool by identifying all possible use cases. These use cases will serve as the main focus when applying and testing the interaction heuristics defined in chapter 3. The next step is to actually test and evaluate the heuristics by using the criteria defined in section 3.2.1. This evaluation may be done by performing TAP sessions, or by performing a user survey. How to do this is described in section 3.3. The resulting scores can be calculated into a grade for the modeling tool, which can then be used to compare the modeling tool to other tools (for the same modeling method).

2. How can we use the evaluation method devised in question 1 to analyse CaseTalk?

   By developing and performing a TAP session using the method described in the previous research question, we can get a number of grades for CaseTalk. These grades can then be averaged to get one single (and final) grade for CaseTalk. The results from the TAP session also indicate some flaws in CaseTalk's design as I have observed in section 5.2.

Bibliography

[1] Activiteiten lectoraat data architectures en metadata management. URL `http://www.han.nl/start/graduate-school/onderzoek/lectoraten-kenniskringen/data-architectures-metadata-management/lectoraat/_attachments/activiteiten_lectoraat_data_architectures_metadata_management.pdf`.

[2] Guido Bakema, Jan Pieter Zwart, and Harm van der Lek. *Volledig Communicatiegeoriënteerde Informatiemodellering FCO-IM*. Sdu Uitgevers bv, Den Haag, 2009.

[3] P.J.M. Frederiks and Th.P. van der Weide. Information modeling: The process and the required competencies of its participants. *Data & Knowledge Engineering*, 58:4 – 20, 2006.

[4] T Halpin. A fact-oriented approach to schema transformation. *Lecture Notes in Computer Science*, 495:342–356, 1991.

[5] T. Halpin. *Information Modelling and Relational Databases, from conceptual analysis to logical design*. Morgan Kaufmann Publishers, 2001.

[6] T.A. Halpin and G.M. Nijssen. *Conceptual Schema and Relational Database design*. Prentice Hall, 1989.

[7] H. Hart, H. Boeije, and J. Hox. *Onderzoeksmethoden*. Boom Uitgeverij, 2007.

[8] S. Hoppenbrouwers. Onderzoeksmethoden 2/best practices/thinkaloud, 1 2009. URL `https://lab.cs.ru.nl/algemeen/Onderzoeksmethoden_2/best_practices/thinkaloud`.

[9] S.J.B.A. (Stijn) Hoppenbrouwers and P.J.F. (Peter) Lucas. Attacking the knowledge acquisition bottleneck through games-for-modelling. In *Proceedings of AISB'09 workshop "AI and Games"*, 2009.

[10] Stijn Hoppenbrouwers, Patrick van Bommel, and Aki Järvinen. Method engineering as game design: an emerging hci perspective on methods and case tools. In *Proceedings of EMMSAD'08 (Exploring Modelling Methods for System Analysis and Design), held in conjunction with CAiSE'08*, 2008.

[11] D Kulak and E Guiney. *Use Cases: Requirements in Context*. Addison Wesley, 2000.

[12] J. Nielsen. Ten usability heuristics, 2007. URL `http://phillips.rmc.ca/courses/459-2007/lectures/03-heuristic-list.pdf`.

[13] Bart Schotten. Designing a game for basic process model elicitation. Master's thesis, Radboud University Nijmegen, 2009.

[14] P. (Patrick) van Bommel, S.J.B.A. (Stijn) Hoppenbrouwers, H.A. (Erik) Proper, and J. (Jeroen) Roelofs. *Innovations in Information Systems Modelling, Methods and Best Practices; Advances in Database Research series*, chapter Concepts and Strategies for Quality of Modeling, pages 167 – 189. IGI Global Publishing, 2008.

[15] Maarten W. van Someren, Yvonne F. Barnard, and Jacobijn A.C. Sandberg. *The Think Aloud Method*. Academic Press, London, 1994.

[16] Ilona Wilmont. A gaming approach to collaborative modelling. Master's thesis, Radboud University Nijmegen, 2009.