Vrije Universiteit Brussel

FACULTY OF SCIENCE AND BIO-ENGINEERING SCIENCE
DEPARTMENT COMPUTER SCIENCE

Dialogue Mapping for GuideaMaps

Master thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in de Toegepaste Informatica

Nick Van Isterdael

Promoter: Prof. Dr. Olga De Troyer
Advisor: Dr. Christophe Debruyne

Academic year 2014-2015

Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
VAKGROEP COMPUTERWETENSCHAPPEN

Dialogue Mapping for GuideaMaps

Masterproef ingediend in gedeeltelijke vervulling van de eisen voor het behalen van de graad Master of Science in de Toegepaste Informatica

Nick Van Isterdael

Promotor: Prof. Dr. Olga De Troyer
Begeleider: Dr. Christophe Debruyne

Academiejaar 2014-2015

Abstract

In any software development process, one of the first phases includes collecting and formulating the requirements that the software should meet. This is important because starting from the right requirements will allow developing software that satisfies the users as well as the customer. This is why collecting requirements is so vital in the whole software development process. In addition, maintaining or modifying existing software can also be a difficult task, especially when it is not clear what the original requirements and rationale were behind software design decisions. Therefore, requirement collection and analysis is an important phase in the development of software.

Both tasks, developing and maintaining software, can be made easier by providing the developers with the design rationale of the original software in an explicit way. Design rationale documentation can provide an insight into why design decisions, such as requirements, have been made, what other alternatives have been considered and why these alternatives have been accepted or declined. Therefore it is important to provide developers with tools to document the requirements collection and analysis process.

In this thesis, I will extend an existing tool that has been created at the Web & Information Systems Engineering (WISE) laboratory of the Vrije Universiteit Brussel. The software tool is called "GuideaMaps" and has been developed to support the requirement elicitation phase for the development of domain specific software.

GuideaMaps has been extended with a dialogue mapping technique. Dialogue mapping is a technique that is used to capture the rationale behind decision-making. It does this by creating a visual and structured representation of a group discussion. This visual representation is called a "decision map". Dialogue mapping takes use of our human ability to grasp and structure visual representation of information easily.

The tool is evaluated by means of a case study involving seven groups of participants with a different technical background. This was done because one of the requirements of the tool was that different stakeholders with a different technical background should be able to use the tool. The results of the study showed that the participants were motivated when using the tool and that the tool was usable by the different stakeholders that participate in the requirement elicitation process. The importance of guidance during the requirement elicitation process and the visual representation of the decision-making process are also discussed.

Keywords: Requirement analysis software, Dialogue mapping, Collaborative decision-making, iOS, GuideaMaps

Abstract

In het software ontwikkelingsproces is een van de eerste fasen het verzamelen en formuleren van de vereisten waaraan de software moet voldoen. Dit is belangrijk omdat starten vanuit de juiste vereisten ervoor zal zorgen dat de software voldoet aan de wensen van zowel de gebruikers als van de klant. Dit is de reden waarom het verzamelen van de vereisten een belangrijke stap is in het gehele software ontwikkelingsproces. Daarnaast kan het onderhouden of aanpassen van bestaande software een moeilijke taak zijn, vooral als de oorspronkelijke vereisten en beweegredenen achter de beslissingen van een software ontwerp niet duidelijk zijn. Daarom is het verzamelen en analyseren van de vereisten een belangrijke stap in de ontwikkeling van software.

Beide taken, het ontwikkelen en onderhouden van software, kunnen gemakkelijker worden gemaakt door de ontwikkelaars te voorzien van de originele motivering achter beslissingen op een expliciete manier. Documentatie over de originele motivering kan een inzicht geven in waarom bepaalde ontwerp beslissingen gemaakt zijn. Dit kan ook een inzicht geven in de alternatieven die overwogen zijn en waarom deze alternatieven zijn geaccepteerd of geweigerd. Daarom is het belangrijk om ontwikkelaars van tools te voorzien die de vereisten voor een software project kunnen verzamelen en de analyse van deze vereisten kan documenteren.

In dit proefschrift wordt een bestaande tool die gemaakt is voor het Web & Informations Systems Engineering (WISE) laboratorium van de Vrije Univertiseit Brussel uitgebreid. Deze tool heeft de naam “GuideaMaps” en was origineel ontwikkeld om de analyse van vereisten voor domain specific software te ondersteunen.

GuideaMaps is uitgebreid met een dialogue mapping techniek. Dialogue mapping is een techniek die gebruikt wordt om de motivering achter beslissingen te documenteren. Dit gebeurd aan de hand van een visuele en gestructureerde weergave van een groepsgesprek. Deze visuele weergave wordt een “decision map” genoemd. Dialogue mapping maakt gebruik van onze menselijke capaciteiten om visuele informatie makkelijk te structureren en te begrijpen.

De tool werd geëvalueerd door middel van een case studie met zeven groepen van deelnemers met een verschillende technische achtergrond. Deze groepen hadden een verschillende technische achtergrond omdat één van de vereisten van de tool was dat deze bruikbaar moest zijn door de verschillende stakeholders die kunnen deelnemen aan het vereisten analyseproces. De resultaten van de studie toonden aan dat de deelnemers gemotiveerd waren om de tool te gebruiken en dat de tool bruikbaar was door de verschillende groepen met een verschillende technische achtergrond. Het belang van

begeleiding tijdens het analyseren van de vereisten en de visuele weergave van de besluitvorming worden ook besproken.

Sleutelwoorden: Requirement analysis software, Dialogue mapping, Collaborative decision-making, iOS, GuideaMaps

Declaration of Originality

I hereby declare that this thesis was entirely my own work and that any additional sources of information have been duly cited. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this thesis has not been submitted for a higher degree to any other University or Institution.

Acknowledgements

First of all, I would like to thank my promoter Prof. Dr. Olga De Troyer and advisor Christophe Debruyne for giving me the opportunity to write this thesis. In addition, I would like to thank them for their help and assistance during the accomplishment of this thesis.

I would like to thank my parents and my girlfriend for supporting my studies and my friends for keeping me motivated.

I would also like to thank the people who participated in the evaluations.

Thank you all.

Nick Van Isterdael
August, 2015

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In any software development process, one of the first phases includes collecting and formulating the requirements that the software should meet. This is important because starting from the right requirements will allow developing software that satisfies the users as well as the customer. This is why collecting requirements is so vital in the whole software development process. In addition, maintaining or modifying existing software can also be a difficult task, especially when it is not clear what the original requirements and rationale were behind software design decisions. These problems become more obvious when the original developers of the software are not around any more or when the software was originally created by a different organization. Therefore, requirement collection and analysis is an important phase in the development of software.

Both tasks, developing and maintaining software, can be made easier by providing the developers with the design rationale of the original software in an explicit way. Although important decisions have to be made during the early phases of requirement collection and analysis, these are also the phases where the least amount of information is available and uncertainty is at its highest. Decision makers are often forced to follow their intuition or experience to base their decisions on. It is thought that these decision-making

processes are usually a rational process in organization, but this is not always the case. Different influences can have an effect on the outcome of a discussion. Decisions are often based on informal reasoning or personal beliefs and other influences such as social pressure can influence the outcome. In most cases, decision-making is done in an unsystematic way and participants fail to document the informal reasoning behind their decisions. Because of this, the rationale behind a decision may remain unknown (Burge & Brown, 2006). Design rationale documentation can provide an insight into why design decisions, such as requirements, have been made, what other alternatives have been considered and why these alternatives have been accepted or declined. Therefore it is important to provide developers with tools to document the requirements collection and analysis process.

In this thesis, I will extend an existing tool that has been created at the Web & Information Systems Engineering (WISE) laboratory of the Vrije Universiteit Brussel. The software tool is called "GuideaMaps" and has been developed to support the requirement elicitation phase for the development of domain specific software.

GuideaMaps was originally developed to support discussions during the requirement elicitation phase of the development of a serious game by providing a list of issues to consider during these discussions (De Troyer & Janssens, 2014). However, the tool is not limited to the development of serious games. It is built in a generic way such that it can be used for each domain for which a list of issues to be considered during requirement elicitation can be provided in advance.

The GuideaMaps tool has been developed as a structured mind-mapping tool, i.e. it uses mind maps that have a predefined structure. This predefined structure is defined by a GuideaTemplate, which models all the important issues to discuss. These issues are presented to the user in a graphical way by creating a map consisting of different "Guideas". A Guidea is a contraction of "idea" and being "guided" A GuideaTemplate is thus a guide for going through the different issues during a discussion.

Although GuideaMaps allow documenting decisions (concerning requirements) taken, it does not allow capturing explicitly the discussions that lead to these agreements, i.e. the arguments used and alternatives considered. The only way to capture a discussion in the original GuideaMaps tool is by adding a comment to the Guidea (i.e. issue) under discussion. This means that the full discussion a group has about a particular issue has to be summarized and saved into a single comment. This has two major disadvantages. First, whether this information is provided is completely dependent on the good will of the user of GuideaMaps. And secondly, summaries lack struc-

ture, which prevents easy processing of this information. Next to the final requirements, it is also useful to document the decision process, as this is the kind of information that will be particularly useful when maintaining the software.

Therefore, GuideaMaps has been extended with a dialogue mapping technique. Dialogue mapping is a technique that is used to capture the rationale behind decision-making. It does this by creating a visual and structured representation of a group discussion. This visual representation is called a "decision map". Dialogue mapping takes use of our human ability to grasp and structure visual representation of information easily and it is especially well suited for complex or "wicked" problems (J. Conklin & Begeman, 1989). Dialogue mapping also addresses some problems that can occur in conventional meetings such as peer pressure or different interpretations of the problem by different stakeholders.

The dialogue mapping is done using a graphical language, IBIS (Kunz & Rittel, 1970). IBIS, which stands for Issue Based Information System, provides a "grammar" to capture the decision-making process and is able to visualize a discussion in a decision map.

The extended GuideaMaps tool can be used in a meeting setting. It will provide all participants with a visual overview of the full discussion. This will support the participants during the meeting to progress in a structured way, but it also makes it possible to use it later to look at the discussed alternatives for an issue (requirement) and recall why they have or have not been accepted. Figure 1.1 shows an example of a decision map made in the extended GuideaMaps tool.

1.2 Research Goals

The goal of this thesis is to extend the original GuideaMaps tool and further support the requirement elicitation process by providing the users of the tool with means to structure and capture the decision-making process during the requirement elicitation phase of a project, i.e. the rationale behind decisions that have been made. In addition, it is important that the tool is still usable by its original target audience: technical as well as non-technical people.

Therefore, the research question that is investigated in this thesis is: "How can we capture the rationale behind decision-making when using the GuideaMaps tool?" This thesis will propose a solution to this problem.

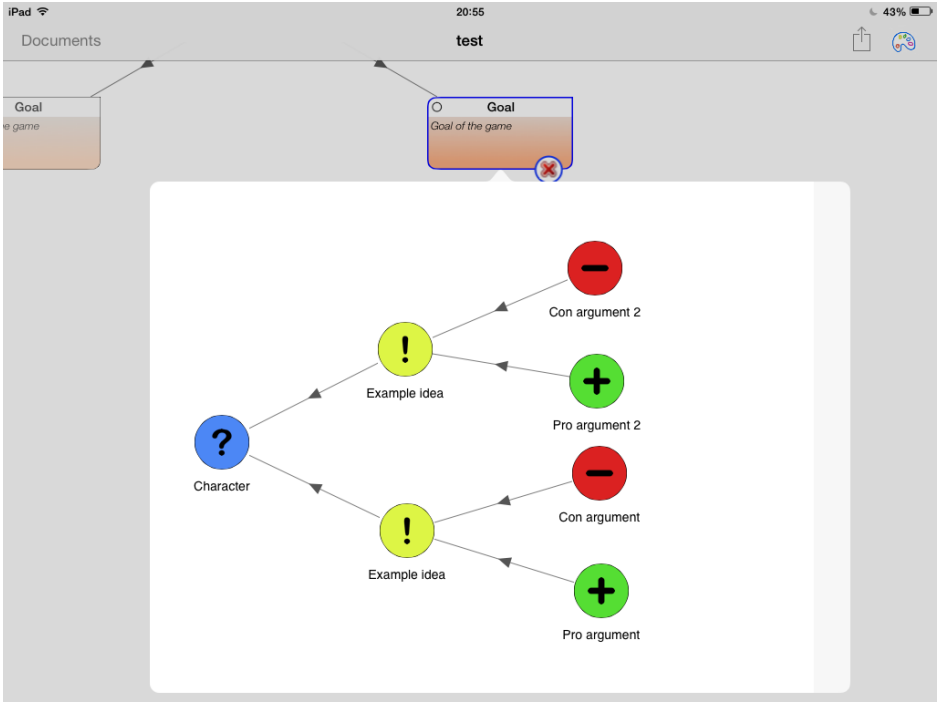

Figure 1.1: The extended GuideaMaps tool

1.3 Research Methodology

To reach the described research goals, a research methodology based on Design Science was used (Peffers et al., 2006), (Takeda, Veerkamp, & Yoshikawa, 1990). This research methodology consists of five phases shown in Figure 1.2. I explain briefly how I performed the different phases.

1.3.1 Awareness of the Problem

To get more insight into the problem and to come up with a proposal, a literature study has been performed to get familiar with concepts such as group decision-making, capturing design rationale, and dialogue mapping. Different graphical languages that can be used for mapping dialogues have been investigated. This was done to choose the most appropriate for the extended GuideaMaps tool. Finally, existing tools and techniques that have been developed to try and tackle the problem of capturing design rationale have been studied. The results of this literature study are described in Chapter 2 and 3.

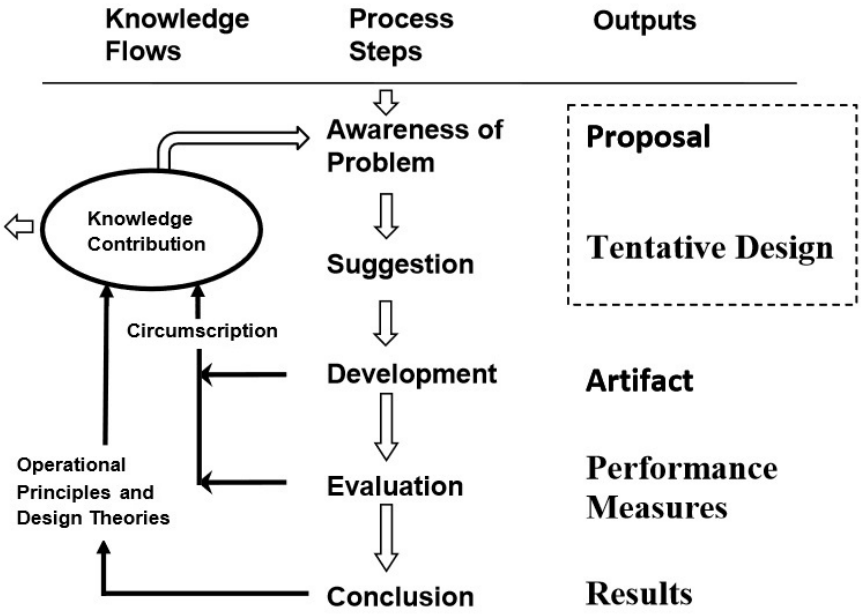

Figure 1.2: Design Science Research Process Model

1.3.2 Solution

The objective was to develop an extension of the original GuideaMaps tool that is able to capture and document the rationale behind decision-making. Based on the research performed in the previous step, I decided to base the design for the extended tool on a dialogue mapping technique and the IBIS framework as the graphical language for capturing design rationale. The motivation for using these techniques as well as the solution is described in Chapter 4.

1.3.3 Design and Development

Before the development of the extended GuideaMaps tool was started, the implementation and functionality of the original tool was analyzed. The original GuideaMaps is implemented as an iOS tablet application. Therefore, it was necessary to get familiar with the Objective-C programming language that was used to develop the original tool and the Xcode integrated development environment needed to develop iOS applications. After these steps, a design and an implementation for the extended GuideaMaps tool was made. This includes the creation of a paper prototype to try out the visualization and the functionalities of the extended tool. The analysis, design and development are described in Chapters 5.

1.3.4 Evaluation

In order to verify whether the extended tool satisfies the original goals, an evaluation was done. To evaluate the extended GuideaMaps tool, first an evaluation methodology needed to be designed. I decided to put the focus of the evaluation on the usability and learnability of the tool given its target audience. As the tool is intended to be used in meetings, groups of at least three participants were recruited for the evaluation. Half of the participants recruited had a technical background while the other half did not have a technical background. This was done to test whether there would be a difference between these two groups of participants. The evaluation itself was based on a use case scenario in which the participants used the tool. After the usage of the tool, the participants were given a questionnaire. Two questionnaires were used. The first questionnaire was designed to evaluate the ease of use, usability and learnability of the tool while the second questionnaire was used to collect the opinion of the participants about the tool. The evaluation and its results are described in Chapter 6.

1.3.5 Conclusion

The conclusions of the work are formulated in Chapter 7. In this final chapter, an overview of the work done in this thesis is given together with a reflection on the results of the evaluation. The limitations of the extended GuideaMaps tool are discussed and some ideas are brought up on what can be done about them in future work.

1.4 Outline of the Dissertation

The subject, motivation, the research goals and challenges of this thesis were introduced in this chapter. Chapter 2 contains the background information needed to understand the rest of this thesis. An introduction to group decision-making, dialogue mapping and the IBIS framework are given. Chapter 3 gives an overview of related work. Chapter 4 gives an introduction to the original GuideaMaps tool and argues why introducing a dialogue mapping technique using the IBIS framework as a graphical language into the GuideaMaps tool can solve the problems discussed in the introduction. Chapter 5 contains the analysis done to start the development of the extended GuideaMaps tool together with the design and implementation of the tool. Chapter 6 contains the evaluation of the extended GuideaMaps tool, the results of this evaluation, and a discussion about the limitations of the

extended tool. The final chapter concludes this thesis by giving an overview of the work done, together with possible future work.

2 Background

In this chapter, I provide the background of the thesis, i.e. information needed to understand the actual thesis work. I will first introduce the reader to group decision-making techniques. These are techniques that can be used by a group of individuals to come to a conclusion about a certain question or problem. Next I will talk about a technique called dialogue mapping. This technique can be used to further enhance the process of group decision-making.

One of the requirements of dialogue mapping is a graphical language. Therefore, I will provide the reader with an introduction to the IBIS method. IBIS is used to capture a discussion in a graphical way.

I will also provide the reader with an example of usage of the IBIS method and an overview of the history of the method.

2.1 Group Decision-Making

Group decision-making or collaborative decision-making is the process where different individuals work together as a group to solve a question or problem. Different factors can influence the outcome of the discussion in a group. Social influence such as peer pressure or group dynamics will have an effect on the final outcome. An accepted outcome can either occur by a group

consensus or by the majority rule approach. (Davis, Brandstätter, & Stocker-Kreichgauer, 1982)

Different techniques have been developed over the years to try and make the process of group decision-making more efficient. The main techniques are:

- **Brainstorming**. During a brainstorming session, participants will spontaneously contribute ideas. The focus is on the generating of ideas, not on the evaluation of ideas. The idea is that in a big collection of ideas, there will be at least one that is interesting. Brainstorming is effective when the problem is simple, for more complex problems, the problem can be split into several simple sub problems. (Osborn, 1953)

- **Nominal group technique**.This technique is similar to brainstorming, but it is more structured and takes a different approach. Every member participating in the discussion will write down his or her own ideas. Social interactions within the group are not allowed as to prevent social influence becoming a factor. A group leader will collect the different ideas and these will be discussed in a group. Every participant will get the chance to defend his or her idea. Participants will score each of the ideas, and in the end the idea with the highest rank will win. (Deip, Thesen, Motiwalla, & Seshardi, 1977)

- **Delphi method**. The Delphi method resembles the nominal group technique but has some key differences. The participants will be a panel of selected experts on the topic. These experts will all receive a questionnaire about the issue and will solve this questionnaire without knowing about each other. A group leader will collect these different questionnaires and create a second round of questions based on the results. This process will continue until a consensus has been found. This technique can be adapted to be used in face-to-face meetings and is then called the mini-Delphi or Estimate-Talk-Estimate. (Linstone, Turoff, et al., 1975)

Regardless of which decision-making method is chosen, it is possible to include this method in a technique called dialogue mapping to keep track of discussions and decisions (J. Conklin, 2005). Dialogue mapping combined with a decision-making technique will enhance the process of group decision-making compared to using an informal method for decision-making.

2.1.1 Dialogue Mapping

Dialogue mapping is a technique for keeping track of a discussion and making decisions about design problems. Three things are required for dialogue mapping: a dialogue mapper, a collaborative display and a graphical language such as the Issue-Based Information Systems method (J. Conklin, 2005). The collaborative display could be a computer screen, a piece of paper, an iPad or any other type of display. A dialogue mapper is a person who will keep track if the discussion (i.e. maps the dialogue) using the aforementioned graphical language.

Dialogue mapping involves creating a diagram of the whole process, starting from the initial problem definition to the decision making process and the final outcome. It differs from informal decision making by providing a structure and a graphical representation of the whole process. It is especially well suited for collaborative decision-making in a group consisting of different stakeholders.

Dialogue mapping tries to improve the decision-making process by providing an overview of the full discussion at all times. As the discussion goes on, the map will grow and will serve as a group memory. This way, participants do not need to keep the whole discussion in their heads. It improves the humans memory capabilities.

It also makes good use of our human ability to grasp visual and structured representations of information faster than textual or unstructured information and makes the process of decision-making easier, more reliable and traceable. Participants can make a more complete and transparent analysis.

2.2 The IBIS Framework

Issue-Based Information Systems, or IBIS for short, is a method developed by Werner Kunz and Horst Rittel in the early 1970s (Kunz & Rittel, 1970). They describe it as a method that can be used to guide the decision-making process for political discussions. According to their paper, IBIS guides the identification of issues, structuring of the discussion and settling of issues. Their original method is focused on the collaborative approach to solving a complex problem in a decision-making group.

The IBIS notation consists of the following three elements:

1. **Issues**: these are the issues that are being discussed by the decision-making group. Typically they are phrased as questions along the lines of "what will we do about x?", where x is the issue being discussed.

2. **Ideas**: ideas are proposed positions or solutions to the issues being discussed.

3. **Arguments**: Arguments can be used in defence (pro) or against (con) a certain position.

Arrows are used to relate these elements to construct a tree.
The IBIS grammar consists of the following three simple rules.

1. An issue can arise from an existing issue or idea. An issue can also arise from an argument. This means that an issue can arise from any other element in the IBIS notation.

2. An idea can only arise from an existing issue.

3. A pro or con argument can only arise from an existing idea.

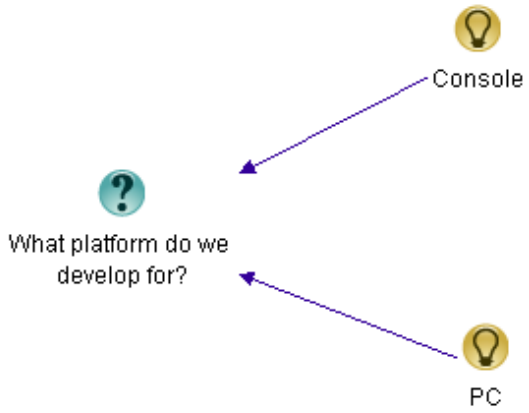

Figure 2.1: Capturing a discussion using IBIS part 1

These simple rules can easily be explained by giving a real life example. Imagine you are running a company that is about to develop a new game. But before you start the development, you have a few questions to answer. I can use the IBIS framework to answer these questions.

An issue that has to be discussed before development could be "What platform do we develop for?" Participants could bring up different ideas like

PC or console for example. So we can start by creating an IBIS graph with an issue and two idea elements. This is shown in Figure 2.1.

Next, the participants can come up with different pro and con arguments for either of these ideas. The pro and con argument elements can be added to the graph and the group can decide to dig deeper into a single idea. This is shown in Figure 2.2.

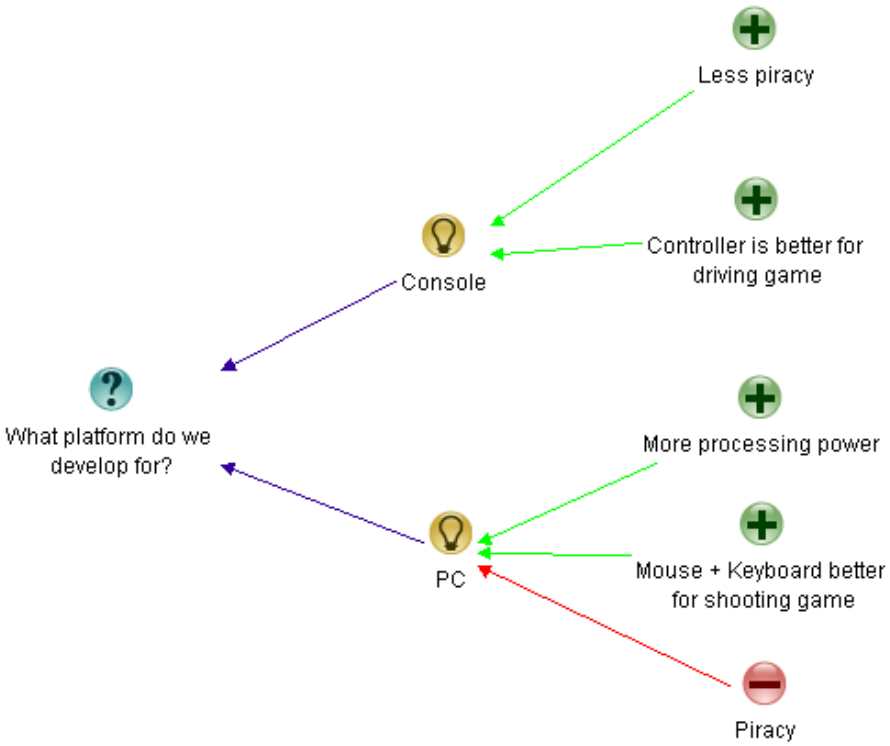

Figure 2.2: Capturing a discussion using IBIS part 2

After the group decides to go through with the console development, they can bring up a new issue. "What console do we develop for?" They can now, as the Figure below depicts, bring up some new ideas such as XBOX ONE and PS4. They can also bring up a new issue from an argument element or any other element. This discussion can go on until they resolve all the issues they encounter and can decide on a final outcome. This is shown in Figure 2.3.

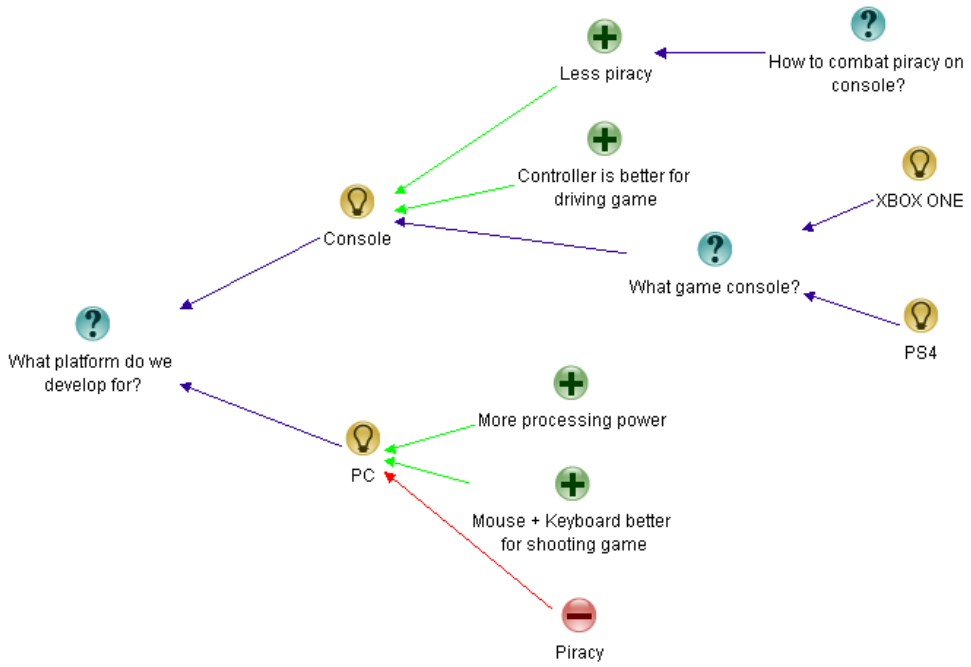

Figure 2.3: Capturing a discussion using IBIS part 3

2.2.1 The Evolution of IBIS

The original method as described by Werner Kunz and Horst Rittel during the 1970s in their paper (Kunz & Rittel, 1970) was supposed to be used and documented on paper. They do talk about a computerized version in their conclusion and state that they started with the first steps of the development.

In 1991, Ray McCall - who was a student of Rittel - introduced an extended version of the IBIS method called Procedural Hierarchy of Issues or PHI (McCall, 1991). This method adds more functionality to the method but also makes it more complex than the original method.

In 1988, Conklin and Begeman describe a first graphical and hypertext-based version of IBIS in their paper called gIBIS: A hypertext tool for exploratory policy discussion (J. Conklin & Begeman, 1988). They make use of colors, hypertext and a relational database to create a first digital tool for the IBIS method.

The original version of IBIS focused on political discussions, this version focuses more on design rational in general. During the 80's, there was a lot of research being done on the topic of design rationale and at the same time new technologies were emerging. Researchers were hopeful that these new

emerging technologies would make the process of capturing design rational an efficient process. They were hopeful that design rational tools would become a widespread and adopted technique in industry and business. (J. Conklin, Selvin, Shum, & Sierhuis, 2001)

But design rationale tools like QuestMap (J. Conklin, 2003) and gIBIS did not become a great success. Although, gIBIS was reported to have some modest success stories (S. B. Shum & Hammond, 1994), (E. J. Conklin & Yakemovic, 1991), (VanLehn, 1985) there was never a widespread adoption. Users did not see the cost-benefit trade-off and it was reported that there was too much cognitive overhead when using these tools (S. J. B. Shum et al., 2006).

Back then, researchers focused on the long-term benefits. They thought that design rational would be something reusable, different people could for example use and share the same design rational, and the design rational would change over time together with the system or problem they are working on. All researchers had to do was solving the capturing problem. But this had as an effect that users did not see the short-term benefits. (J. Conklin et al., 2001)

Researchers had learned a lot from their earlier failures and were still encouraged by the limited success of gIBIS and QuestMap. They were confident that the problems were surmountable. This gave rise to a new tool called Compendium (Okada & BuCkiNGhAm Shum, 2006). Compendium also used the IBIS method to capture design rational. The focus was on the long term but also the short-term benefits of capturing design rational. The tool could be used by an individual or in a group context.

IBIS has been an inspiration for different software tools that can be used for capturing discussions or design rational. It has also been an inspiration for different techniques or methods to capture design rational. I will discuss some of these tools and techniques in the related work section.

2.3 Conclusion

In this chapter, I have presented different techniques that can be used to facilitate the collaborative discussion-making process. I discussed into more detail a technique called dialogue mapping that can be used to further enhance this process.

Dialogue mapping requires three things: a shared display, a graphical language to capture the discussion, and a person who will document the discussion making process, i.e. the dialogue mapper (J. Conklin, 2005).

For the graphical language needed for capturing the discussion I have

presented the IBIS technique. This is a simple and intuitive technique that is powerful enough to capture the discussion about any possible design problem. I have given an example of use of the IBIS method, an overview of the history of the method and the advantages of the method.

3
Related Work

In this section, I will give an overview of different models that can be used to capture design rationale. Each of these models is useful in a certain situation or with a certain goal in mind. One method can for example only be used for the argumentation of one specific issue, while another method can express the relationship between different issues. I will also compare these different models with the IBIS method, and explain why I chose to go with the IBIS method.

Next, I will give an overview of different software tools that have been developed to support the capturing of design rationale. Again, these different tools are useful in different situations.

Finally, I will review research done about using a collaborative display in a group decision-making setting.

3.1 Argumentation Based Models

IBIS is not the only model that is able to capture design rationale. Many models exist. Some of these models are: the Toulmin model (Toulmin, 2003), Procedural hierarchy of issues (McCall, 1991), Questions Options and Criteria (MacLean, Young, Bellotti, & Moran, 1991), Decision Representation Language (Lee, 1989) and RATSpeak (Burge & Brown, 2004).

3.1.1 Toulmin Model of Argumentation

The Toulmin model of argumentation is the earliest argumentation based model used for design rational that I encountered in the literature. It was developed by Stephen Toulmin in the late 1960's and proposed in his book: The uses of argument (Toulmin, 2003). Toulmin noticed that arguments typically consist of these 6 parts:

- Claim: a statement that you ask the other person to accept.
- Ground: the basis of persuasion, which is made up of data or facts.
- Warrant: links the ground to a claim.
- Backing: gives additional support to the warrant.
- Modality: indicates the strength of the leap from the ground to the warrant.
- Rebuttal: Counter-arguments or statements indicating circumstances when the general argument does not hold true.

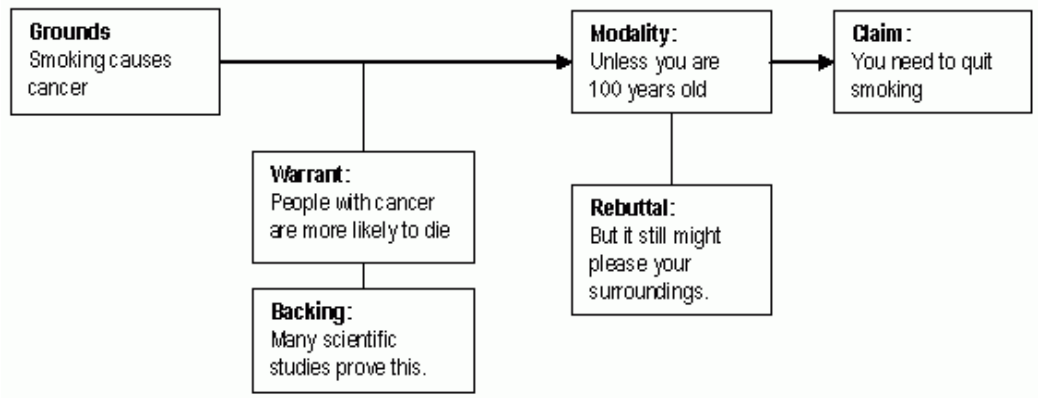

Figure 3.1: The Toulmin Model of Argumentation

The Toulmin model can only look at a single case. Relationships between different claims cannot be represented in the original method. It is solely a model of argumentation. To support relationships between claims, the model should be extended.

3.1.2 Questions, Options, and Criteria

Design space analysis is an approach to capturing design rationale which uses a semiformal notation called Questions, Options and Criteria, or QOC in short (MacLean et al., 1991). QOC uses the following basic building blocks:

- Questions: for identifying issues.
- Opinions: are the possible answers or solutions to the issue.
- Criteria: the bases for choosing among the options.
- Assessments: whether an opinion supports or challenges a criteria.

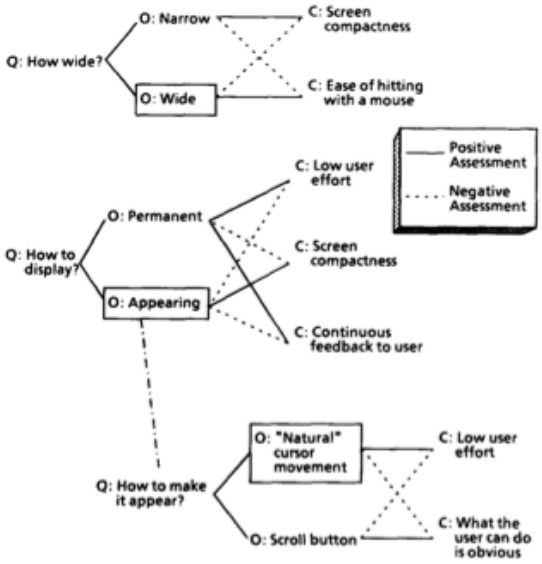

Figure 3.2: Questions, Options and Criteria

3.1.3 Procedural Hierarchy of Issues

Procedural Hierarchy of Issues (McCall, 1991) is a variation on IBIS that extends the original IBIS notation. It was developed by Ray McCall who was a student of Rittel. The difference is that it changes the structure in which issues, ideas and arguments are related and puts more emphasizes on the relationship between issues. In PHI, an issue will "serve" another issue. There will be a relationship of issues and sub issues and this will

create a tree structure of issues. There can also be sub answers or sub argument relationships.(Jarczyk, Löffler, & Shipman III, 1992) JANUS and MIKROPLIS (McCall, 1989) are two software tools that use this modified version of the IBIS method.

In GuideaMaps relationships between issues are already captured to a certain extend by means of the hierarchical decomposition of the Guideas and by the dependencies between Guideas ("requires" and "excludes"). Therefore, the extension proposed by PHI is not needed for my purpose.

3.1.4 Decision Representation Language

Decision Representation Language (DRL) is another method that can be used to capture design rationale, but this method focuses more on the decision making process. It can be used to represent aspects of decision making such as the alternatives being evaluated, goals to satisfy and the arguments evaluating the alternatives. DRL also provides a vocabulary including issues, pro and con arguments and relationships among alternatives. Once DRL captures the decision process, it can provides services to help with the decision making process. The goal of this method is to provide knowledge sharing, decision support and problem solving. According to a paper published by Lee J,RDL is more expressive than other languages (Lee & Lai, 1991). But by making this method more expressive, it will also become more complex and less intuitive than the IBIS method. Figure 3.3 shows an example of DRL.

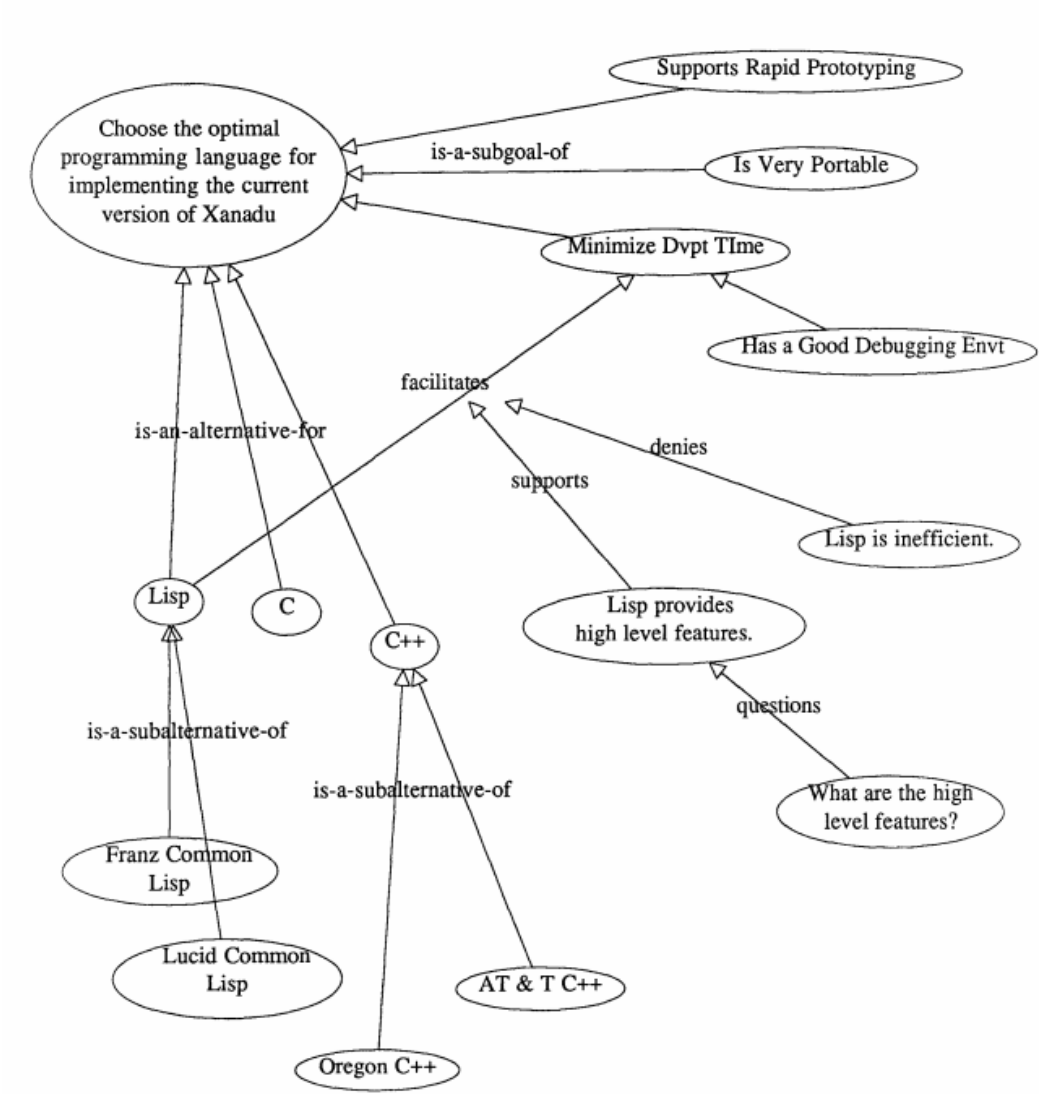

Figure 3.3: Decision Representation Language

3.1.5 RATSpeak

RATSpeak was developed by Burge and Brown and is based on RDL (Burge & Brown, 2004). RATSpeak extends RDL by adding different types of arguments. In RDL there are no means of saying an argument is for or against a position. Figure 3.4 shows the argumentation structure used in RATSpeak. Alternatives for each decisions-problem can be argued by their relationship to requirements, assumptions or claims that support or deny an alternative. Decisions can be divided into sub-decisions and an alternative can lead to a new decision problem.

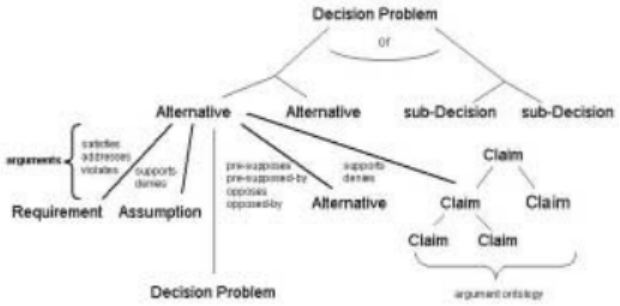

Figure 3.4: RATSpeak argumentation structure

Some of these models I have presented are more expressive than the IBIS method, but with more expressive power comes more complexity. From earlier research, as I have stated in the background section, it is known that more expressive power does not equal more efficiency. There is a fine line between offering the user more functionality or expressive power and the ease of use or willingness to use a method or software tool implementing this method. (Jarczyk et al., 1992)

3.2 Shared Display for Decision-Making

I stated earlier that one of the three requirements for dialogue mapping was a shared display. This shared display is used to create a mental model of the discussion. This way all the participants in the decision-making group have an overview of the discussion so far.

Tablets are a huge success on the consumer market. And according to a study done by Hess and Jung in 2012 tablets can also add value to the business environment (Hess & Jung, 2012). More and more companies are

starting to use tablets for business. This can either be for personal use, or for collaborative use. A tablet can for example be used to give a presentation. It can be a low cost alternative for people who do not need all the functionalities of a laptop. Because of the growing success of tablets in business, it can be an interesting idea to create collaborative tools for these tablets.

Bolstad and Endsley in 1999 (Bolstad & Endsley, 1999) experimentally studied the usefulness of a shared display in a collaborative setting. They concluded that using a shared display could enhance the effective performance of the team. However, they also concluded that a shared display could decrease the performance. The key is to provide just enough information to the group so they can build a mental model of each participants' opinion or goals. Providing too much information can have a negative effect on the group performance.

A study done by Andriy Pavlovych and Wolfgang Stuerzlinger (Pavlovych & Stuerzlinger, 2008) looks at the performance between different types of shared displays. They conclude that for a shared display oriented vertically, for example on a wall, the interaction speed was significantly faster, by 51%. They also observed that the performance of a group using a shared display scaled almost linearly with the number of participants.

3.3 Software Tools for Decision-Making

3.3.1 gIBIS and rIBIS

gIBIS stands for graphical IBIS and was developed by Conklin and Begeman in 1988 (J. Conklin & Begeman, 1988). The tool is hypertext-based and can be used to capture design rational. It provides a graphical representation for the IBIS method. The tool extends the original IBIS method by adding another type of node. This node gives the user the ability to add external material such as text documents. The gIBIS user interface is shown in Figure 3.5.

The user interface primarily exists of 2 parts:

1. The first part is a canvas, in which a user can create different nodes and connections. Colours can be used to indicate the type or state of nodes and connections. There is a global view that shows the entire network and a local view that shows the fine structure of a part of the network.

2. The second part is called the index view and provides the user with an ordered hierarchical view of the nodes in the current network. This is

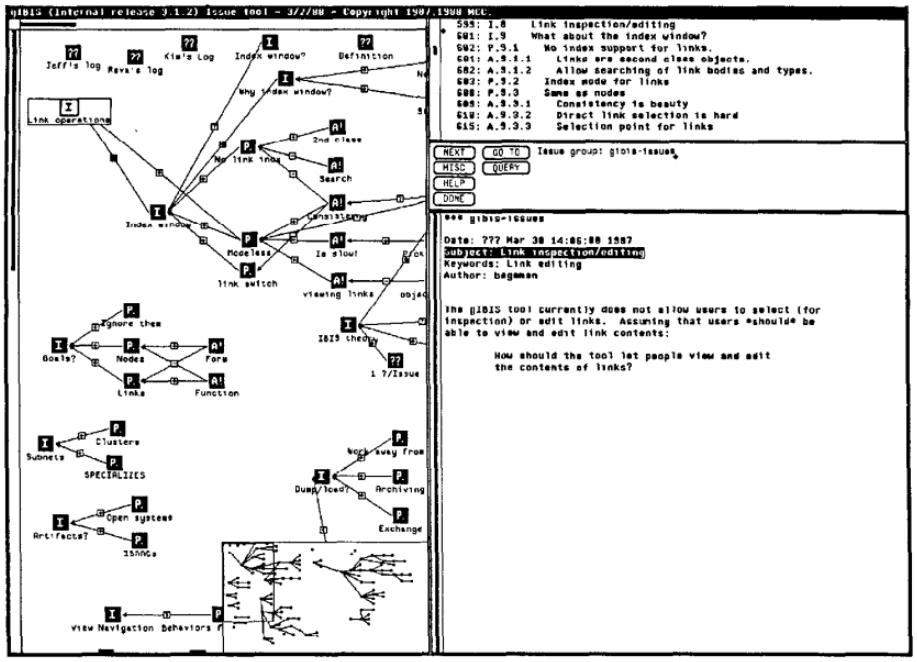

Figure 3.5: The gIBIS user interface

a textual representation of the graphical IBIS network.

rIBIS is an extension of the gIBIS tool, rIBIS stands for real-time IBIS and gives multiple users to ability to browse and edit multiple views of an IBIS network. It offers two types of editing, tightly coupled and loosely coupled. In tightly coupled mode, changes that a user makes are immediately reflected in the global view and to other users. In loosely coupled mode, changes are done in a local and private version and are not immediately visible to other users.

The gIBIS and rIBIS tools never became a great success and Conklin and Begeman also concluded in their paper that there were some short-comings to their tool. They stated that the problems were related to the user interface and to the method of working with the tool. The interface was not supporting an efficient process of capturing design rational. They concluded that there was a delicate balance between the ease of use and the complexity of the IBIS model and method used.

3.4 Conclusion

I have presented different models that can be used to capture design rationale, some of these models are more expressive than others and are suitable for different situations.

I also compared them with the IBIS method. IBIS offers the right balance between expressive capabilities and ease of use. The IBIS notation will not create a structure that becomes too complex for a user to grasp but will at the same time be expressive enough to be able to reason about any design problem.

One could argue that there are different design rationale models that could fulfill my needs if they would be extended with some new functionality. However, it would not be beneficial to extend a model when there is already an existing model providing me with everything I need.

I also reported on the use of a shared display in the context of a group decision making process and found that they could improve the performance and looked into literature about the use of tables in a business environment.

Finally, I investigated software tools that have been used to capture design rationale. Some of these have not been successful, but have given researchers new insights into the process of capturing design rationale and the importance of a good user interface design. I observed that it is important to correctly balance expressive power and ease of use.

4

Dialogue Mapping for GuideaMaps

4.1 Introduction

This chapter will focus on the solution proposed for extending the original GuideaMaps tool with functionality to support and capture decision-making. First I provide an introduction to the original GuideaMaps software tool and a short overview of its functionalities. This chapter will furthermore present some findings that are brought up in the paper presenting GuideaMaps (De Troyer & Janssens, 2014). As stated in Section 1.1, the original tool is not capable of capturing the rationale behind a decision, because the only option for a user to add information about the decision to an issue discussed is by adding a comment. A comment does not provide enough structure to support this. This can be achieved by adding a dialogue mapping technique, which will be presented in Section 4.3.

4.2 GuideaMaps Tool

The GuideaMaps tool has been developed as a structured mind-mapping tool that has a predefined structure (De Troyer & Janssens, 2014). This prede-

fined structure is defined by a GuideaMaps template, which takes all the important issues for creating software in a particular domain into consideration. These issues are presented to the uses in a graphical way by means of a map consisting of different "guideas". A guidea is a combination of both the notion of an "idea" and being "guided". A GuideaMap is thus a guide for going through the different issues during the requirement elicitation.

The original tool is intended to be used for the requirements elicitation for serious games. The tool provides the user with a GuideaMaps template considering all the important issues related to the creation of such a game. New templates can be added to extend the applicability of the tool towards other domains than serious games. For example, a template considering all the issues related to the development of a web application could be added. This makes the tool suitable for a wider variety of fields. Guideas that could be considered for a Web application could be: "Do you want to provide a search functionality?" or "Do you want to secure parts of the website?"

A guidea, which represents an issue that has to be discussed, can be decomposed in sub-guideas. These sub-guideas are connected to its parent by a line and arrow. Not all guideas are mandatory and some guideas can have predefined possible options as answers to the issue being discussed. For example, "PC" or "Console" as possible values for the issue "What platform do we develop for?"

The tool provides an easy to use point, tap and drag user interface developed for use on a tablet. This way, they tried to create a user interface that is usable by any stakeholder participating in the requirement elicitation process. A user can drag the different guideas over the guidea map and collapse or expand guideas and sub-guideas. When a user double taps on a guidea, a popover will appear in which the user can add a comment to the guidea. This comment gives the user the ability to capture what has been discussed during a meeting. Figure 4.1 shows the GuideaMaps user interface.

The authors of the GuideaMaps paper bring up some findings (De Troyer & Janssens, 2014):

- **The requirement analysis phase has to include all the stakeholders relevant to the project**: "In software engineering, it is more and more accepted that the requirement analysis phase should include all the stakeholders relevant to the project in the design process. This will create a more efficient development process and a better end product for the user, thus creating a bigger chance of successfully finishing a project (Muller & Kuhn, 1993). Because of this, the ability of the tool to be used by different stakeholders is an important issue. The different stakeholders usually have a different background and expertise.

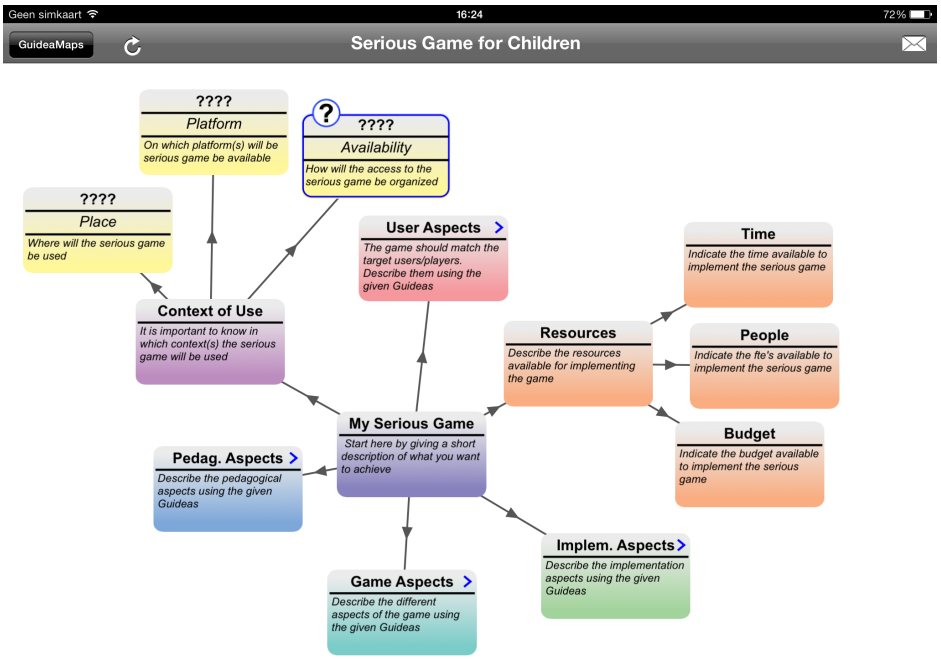

Figure 4.1: The original GuideaMaps tool

The tool should be useable by computer scientists who are experienced in using different software tools, as well as people who have no experience with software tools at all".

- **Guidance is needed during requirement analysis** :"(Muller & Kuhn, 1993) also concluded that some guidance is needed during the meetings. Because the stakeholders are from a different background and have different expertise, the discussion of an issue is not always equally clear to all of the stakeholders involved in the discussion. During a conventional meeting, some aspects that are relevant to the success of the project may not be taken into consideration because of a lack of guidance during the requirement analysis process. Guidance in this process can lead to higher efficiency".

4.3 Decision Mapping for GuideaMaps

In the introduction, I stated that group decision-making is not always a systematic process and that the rationale behind decisions is not always documented. A requirements document captures what decisions are made (i.e.

which requirements are kept) but it does not capture why these decisions are made. Because of this, the rationale behind decisions may remain unknown.

The only way to capture a discussion in the original GuideaMaps tool is by letting a user add a comment to a particular guidea (i.e. issue). This means that the full discussion a group had about one particular issue, has to be summarized and saved into one single comment. This has two major disadvantages. First, whether this information is provided is completely dependent on the good will of the user of GuideaMaps. And secondly, if such information is given, it lacks structure that prevents easy processing of this information. Figure 4.2 shows the screen where a user can add a comment to a guidea.

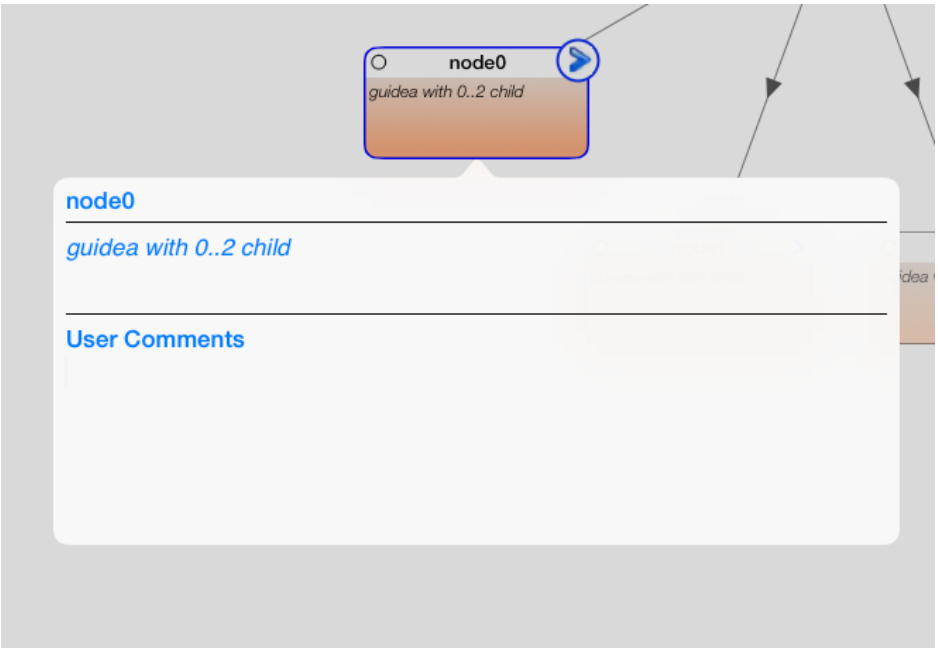

Figure 4.2: The GuideaMaps comment screen

The original GuideaMaps tool provides guidance to the users in dealing with issues to consider by means of a Guidea Template. This template gives an overview of all the relevant aspects that have to be taken into consideration during the decision process and their relationships (by connecting them) and is a starting point for creating the actual GuideaMap for the application.

To improve the decision making process with the GuideaMaps tool, I will introduce a dialogue mapping technique into GuideaMaps. In the background section I stated that dialogue mapping requires three things: a collaborative display, a graphical language, and a dialogue mapper (J. Conklin, 2005).

- The original GuideaMaps tool was developed for a tablet, an iPad to be more specific. A tablet is lightweight and mobile. This makes it an ideal tool to be used in a meeting setting. More and more companies and individuals are adopting this technology (Ozok, Benson, Chakraborty, & Norcio, 2008). An increasing number of people are getting used to the touch interface provided by tablets and report that tablets are fun and easy to use. A tablet can also be connected to a projector and project its screen on a wall. This way, the screen can easily be shared by a group of people participating in a meeting. This makes a tablet an ideal device to be used as a collaborative display in dialogue mapping.

- The second requirement is a graphical language. This graphical language is used to capture a dialogue and create a graphical representation of it. The IBIS method - elaborated on in detail in Chapter 2 - is the graphical language I will adopt because it is simple and intuitive to use and at the same time powerful enough to capture discussions and the rationale behind any possible decision. The notation is powerful because it allows a group of people to structure their discussion or their thoughts by using a very limited number of building blocks. IBIS offers a complete overview of a discussion at any time, which creates a habit of asking the right questions at the right time and makes sure the discussion does not go off track, in this way helping people to reach a conclusion. IBIS also provides just enough structure to keep a diagram growing without it becoming too complex or overwhelmed with information. Years of experience have shown that IBIS does not introduce too much cognitive overhead but has just enough structure to capture a conversation. This makes it ideal for a meeting environment. Furthermore, the concept of answers and questions is something that everyone uses in everyday life. This makes an existing IBIS map easy and intuitive to understand for someone getting involved in the discussion without having to study it first. Furthermore, the IBIS notation consists of three basic elements and three simple rules, this makes it easy for someone to learn and use it. (J. Conklin & Begeman, 1989)

- The third requirement for dialogue mapping is a dialogue mapper. A dialogue mapper is a person who will map out the discussion using the aforementioned graphical language. The dialogue mapper should have knowledge of the graphical language being used, and be able to translate real life scenarios into graphical maps. I also envisage the dialogue mapper taking the lead of the discussion and serve as a central person that steers the conversation, or he can be a normal participant

that only concerns himself with mapping the discussion. By introducing a dialogue mapper that can take the leading role in the group decision-making process, more guidance

The original GuideaMaps tool can only add a comment to an issue. By creating an IBIS map for every issue that has to be taken into consideration by the participants, the rationale behind the decisions for every issue can be captured in a structured way. The dialogue mapper can connect the iPad device to a beamer (or a screen) and guide the group through the decision-making process. The beamer can make the IBIS map visible to all the stakeholders participating and because IBIS is a graphical language that is simple and intuitive to use and read, all stakeholders participating in the decision-making can follow the discussion. An overview of the full discussion will always be available to all the participants. At any moment in time, the participants can look at the alternatives that have been discussed and why they have been accepted or declined.

Humans have a limited short-term memory capacity and thus individuals can only keep a limited amount of ideas in short-term memory (Miller, 1956). By providing a graphical overview of the discussion, the problem of humans limited short-term memory can be addressed. Participants in the discussion might bring up an idea or an argument that has already been brought up before, they might bring up a new argument that is similar to an existing one but is differently worded or a part of the discussion might still need further exploration. These problems can be avoided by providing a graphical overview of the discussion.

Also the problem of "circular logic" can also be addressed. Participants might bring up ideas or arguments that have already been addressed in a different fashion. This might lead to situations where the same ideas or arguments keep on being discussed in a different way. By providing the overview of the discussion, this can be avoided to a certain extend and the discussion can be steered in a forward way.

4.4 Conclusion

I have introduced the reader to the GuideaMaps tool, which is a tool developed to support the requirement elicitation phase for software development. The tool tries to improve the decision making process by providing guidance during the requirement analysis. A Guidea Map provides the important issues that have to be discussed. But the tool does not provide the users with means of capturing the rationale behind decisions taken, except for adding an unstructured comment to an issue.

To solve this problem, I propose to extend the tool with the dialogue mapping technique that can capture the rationale behind a decision. By implementing the IBIS method into the tool, users can capture the rationale behind the decisions made for every issue. Every issue (or guidea) will be given an IBIS map, which provides a mean to document in a graphical way the rationale behind the decision taken for the guidea. These graphical maps will be created by the dialogue mapper, i.e. the person that translates the discussion into the dialogue maps.

The dialogue mapper will need a good understanding of the IBIS method to be able to create the IBIS maps, however the method is simple and does not require a lot of learning time. A dialogue mapper needs to map the conversation while it is flowing, thus needs to be fluent in converting natural conversations into IBIS maps. The role of the mapper can be a leader role or a normal participant role.

5 Development

5.1 Introduction

In chapter 4 I explained how to extend the GuideaMaps tool with the dialogue mapping technique to capture the whole decision-making process in a graphical manner. I adopted the IBIS method for capturing the discussion. I also explained how I imagine the extended tool to be used; a dialogue mapper capturing the discussion using an iPad, which is connected to an external screen that all other participants can see. An iPad is a mobile device offering a touch screen interface that runs on the iOS operating system. This chapter explains the development of the extension. It is split up in three parts, the analysis of the original GuideaMaps tool and the prerequisites needed to develop the extended tool, the design of the extended tool and finally the implementation of the extended tool.

5.2 Analysis

In this section, the iOS operating system will be introduced, together with the Xcode integrated development environment (IDE), the latter being needed to develop an iOS application. The functionalities and inner working of the original GuideaMaps tool are also analyzed.

5.2.1 iOS Platform

The original GuideaMaps tool has been developed for the iOS platform and I will continue the development using this platform. The iOS platform is an ideal platform for developing Apple applications for mobile devices using a touch screen interface because:

- Apple offers a limited range of devices for which can be developed for and the same manufacturer builds them all. They also offer consistency in their operating system versions over their devices. Because of this, building an application or user interface is easier compared to other operating systems such as Android. Many different Android device manufacturers exist and there is OS fragmentation across all of those devices. This makes it harder to create an application or user interface that works on every Android device. The OS fragmentation of Android and iOS devices is shown in Figure 5.1.
- iOS is user interface friendly because the Apple Xcode IDE offers a built-in user interface builder. It also offers frameworks to create a user interface that makes it easier for a developer. Because of this, a developer is more restricted but he does not have to start from scratch or worry about consistency over different devices.

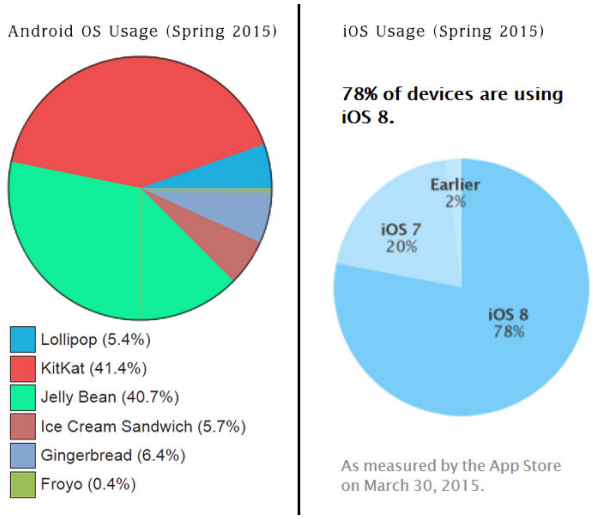

Figure 5.1: OS fragmentation Android and iOS 2015 (*Android and iOS OS fragmentation*, n.d.)

iOS Operating System

The iOS operating system was developed by Apple and it is the operating system used for the iPhone, iPad, iPod touch and Apple TV. iOS is a mobile operating system primarily designed for devices with a touch screen interface. In January 2015, its market share on the mobile market was the second largest with 42.59%, just behind the Android operating system. This is shown in Figure 5.2.

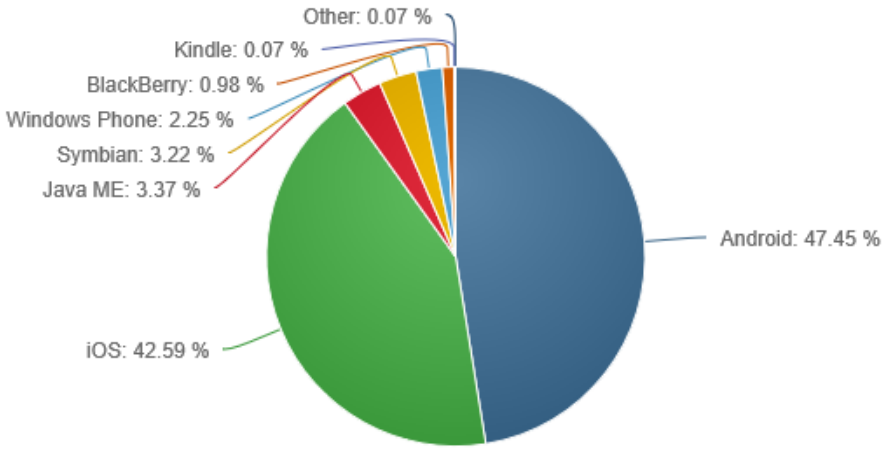

Figure 5.2: iOS market share on the mobile market

iOS was originally developed in 2007 for the iPhone. Over the years, the iOS operating system has been extended to support a wider range of devices, deliver more functionality and create a better user interface. At the moment of writing, the most recent version of iOS is iOS 8. GuideaMaps has been developed for the iOS 7 version and because I have started the development of the extended tool before the iOS 8 update, I have continued the development using iOS 7.

Xcode and iOS SDK

Xcode is an integrated development environment (IDE) developed by Apple. It provides a wide array of developer tools such as a source code editor, a graphical user interface editor, a mobile device simulator, a debugger and many others. It also provides the iOS SDK which contains software development tools and documentation for developing in OS X and iOS. Figure 5.3 shows an overview of the Xcode IDE.

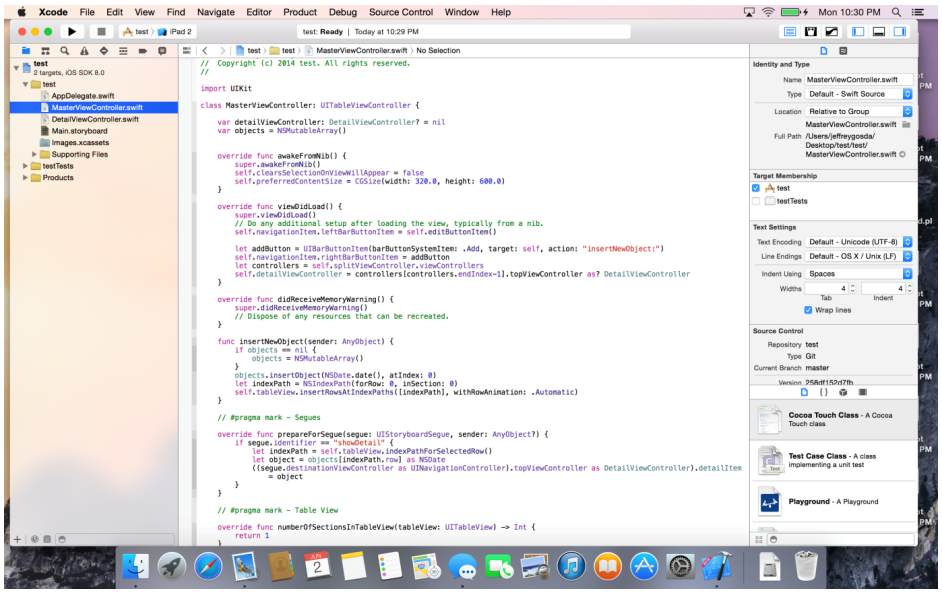

Figure 5.3: The Xcode source code editor

The graphical user interface editor provided with Xcode is called storyboard. A storyboard is a visual representation of the user interface of an iOS application. It shows an overview of all the scenes used in an application and the connections between them. A scene in the storyboard is the combination of a view and its controller. Scenes are connected by "segue" objects that represent transitions between two view controllers.

The storyboard provides a developer with the functionality to add views such as buttons or labels to a scene. The storyboard also allows a developer to connect a view to its controller object and to manage the transfer of date between view controllers. This is shown in Figure 5.4.

Objective-C

Objective-C was developed during the early 1980's by Brad J Cox and is built on top of the C programming language. Until 2014, it was the primary programming language used for developing OSX and iOS applications. As such, the original GuideaMaps was implemented in Objective-C. Objective-C is a superset of the C programming language and provides object-oriented capabilities and a dynamic runtime. Objective-C inherits the syntax, primitive types, and flow control statements of C and adds syntax for defining classes and methods.

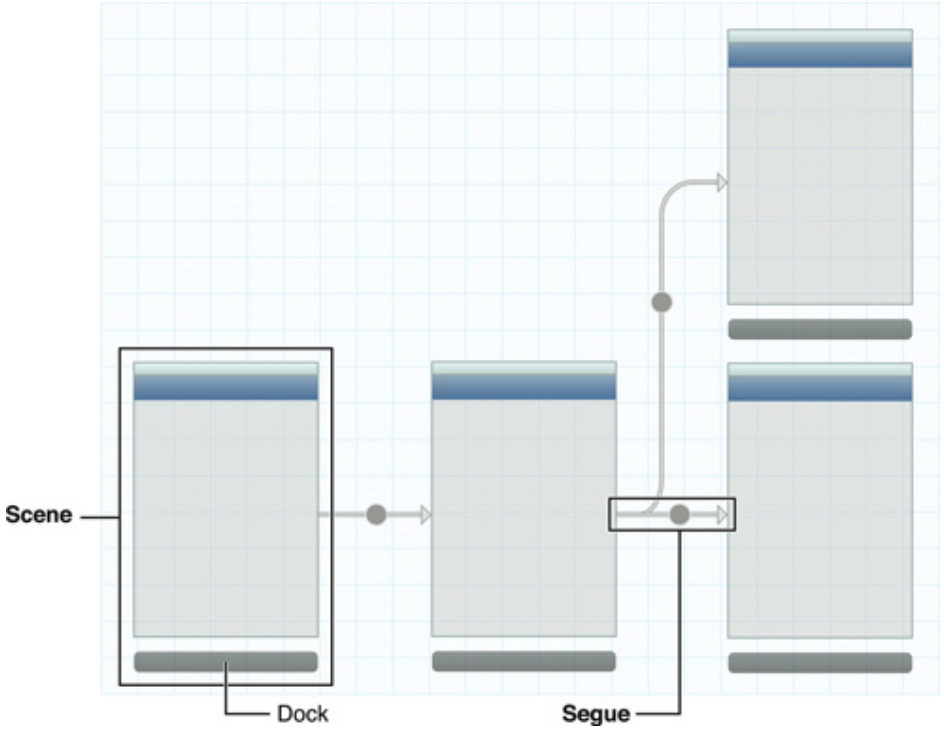

Figure 5.4: Storyboard scenes

5.2.2 GuideaMaps Tool

The GuideaMaps tool provides the user with GuideaTemplates. A GuideaTemplate is a document that contains information about the issues that have to be taken into consideration. It also contains a predefined structure for the GuideaMap and some extra information about the template such as the author of the template and a description.

Different templates can be created for different domains(serious games, web applications, etc.). A template defines the issues that have to be taken into consideration for making an application in a particular domain.

Figure 5.5 shows the screen that is used to create a new GuideaMap. The user has to enter a name for the map and select an existing GuideaTemplate. A GuideaTemplate has an author and a description. This screenshot only contains one GuideaTemplate named "Test Model".

These GuideaTemplates are stored as XML files, which are in turn defined by XML schemas. XML stands for Extensible Markup Language and was designed to describe data by defining a set of rules for encoding documents in a format that is readable by humans as well as machines. The design goals of XML are simplicity, generality and usability across the web and other

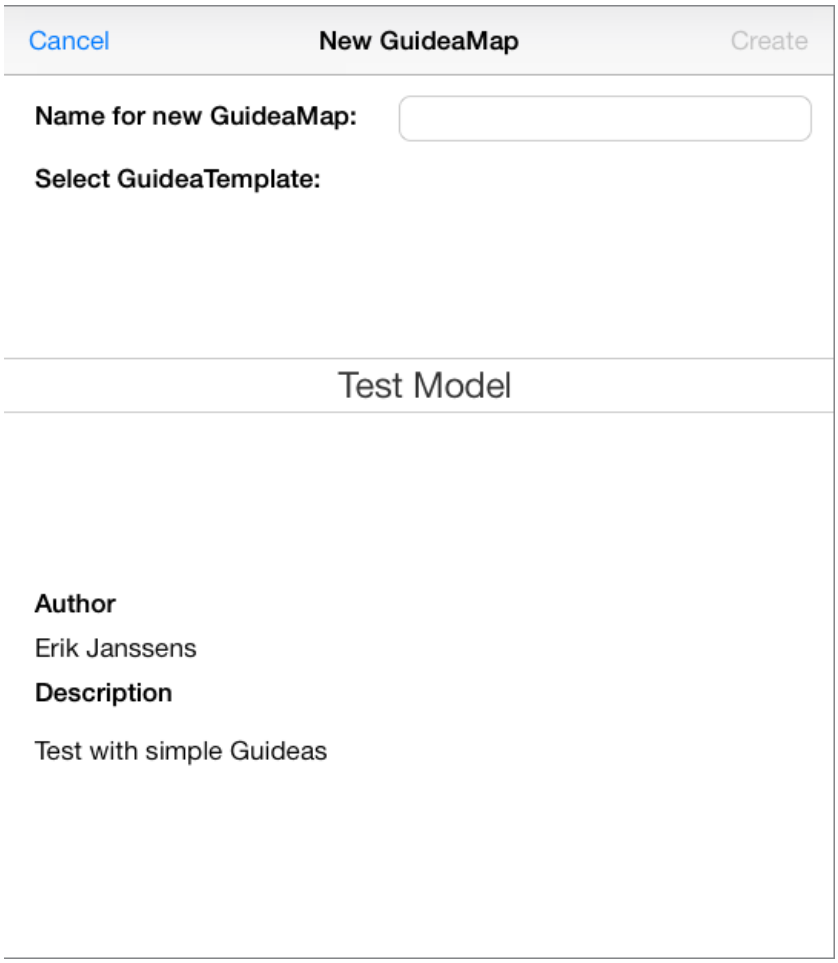

Figure 5.5: The create a new guidea map screen

applications. XML is widely used for the representation of data structures such as those in web services (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 1998).

An XML document can be described by and validated against an XML schema. Typically, an XML schema is a constraint on the structure and contents of an XML document. An XML schema will provide a description of the allowed content of an XML document.

A GuideaTemplate is created as an XML document. The XML document contains the name of the author, a description of the template and the information needed to create all the guideas (or issues) that have to be taken into consideration. The guideas are defined as features in the XML document and I will refer to them as features from now on in this section. The following image shows an example of an XML document used to create

a GuideaTemplate. Figure 5.6 shows an example of a GuideaTemplate.

```
<?xml version="1.0" encoding="UTF-8"?>
- <product xsi:noNamespaceSchemaLocation="http://www.codefromtheattic.com/featureModel.xsd" xmlns:xsi="h
    <model>Serious Game</model>
    <author>Olga De Troyer</author>
    <comment>This model contains all the characteristics for defining a serious game.</comment>
  - <feature id="root">
      <name>Root</name>
      <description>This game will provide different scenarios against cyberbullying</description>
    - <feature id="purpose">
        <name>Purpose</name>
        <description>Pedagogical purpose</description>
        <cardinality max="1" min="0"/>
      </feature>
    - <feature id="goal">
        <name>Goal</name>
        <description>Goal of the game</description>
      </feature>
    - <feature id="char" abstract="TRUE">
        <name>Character</name>
        <description>Game Character</description>
        <cardinality max="2" min="1"/>
      - <feature id="char1" extends="TRUE">
          <name>Hero</name>
          <description>Hero Character</description>
        </feature>
      - <feature id="char2" extends="TRUE">
          <name>Buddy</name>
          <description>Buddy Character</description>
        </feature>
      </feature>
    </feature>
</product>
```

Figure 5.6: GuideaTemplate xml document

This particular GuideaTemplate describes a GuideaMap with 5 features. The features have the following structure:

- Root
  - Purpose
  - Goal
  - Character
    - Game Character
    - Buddy

The template also contains a model name “Serious game”, an author “Olga De Troyer” and a description “This model contains all the characteristics for defining a serious game”.

A GuideaTemplate XML document must be conform to an XML schema, in this case the XML schema featureModel.xsd. This XML schema provides an overview of the structure of a GuideaTemplate XML document. Figure 5.7 shows the XML schema that is used to define a GuideaTemplate XML documents.

```
▼<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified" elementFormDefault="qualified">
  ▼<xs:element name="product">
    ▼<xs:complexType>
      ▼<xs:sequence>
          <xs:element name="model" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="author" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element ref="feature" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="feature" type="featureType"/>
  ▼<xs:complexType name="featureType" mixed="true">
    ▼<xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
      ▼<xs:element name="cardinality" minOccurs="0" maxOccurs="1">
        ▼<xs:complexType>
          ▼<xs:simpleContent>
            ▼<xs:extension base="xs:string">
                <xs:attribute type="xs:string" name="min" use="required"/>
                <xs:attribute type="xs:string" name="max" use="required"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element ref="feature" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element type="xs:string" name="requires" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element type="xs:string" name="excludes" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="id" use="required"/>
      <xs:attribute type="xs:string" name="abstract" use="optional"/>
      <xs:attribute type="xs:string" name="extends" use="optional"/>
      <xs:attribute type="xs:string" name="optional" use="optional"/>
    </xs:complexType>
</xs:schema>
```

Figure 5.7: GuideaTemplate xml schema

I will clarify this XML schema by creating an Object Role Modeling (ORM) diagram from its content. I do this because ORM diagrams are more intuitive than an XML schema.

Information systems are best defined first at the conceptual level using concepts and languages that people can readily understand. ORM simplifies this process by using natural language and intuitive diagrams.

ORM is a method used for modeling and querying an information system at the conceptual level (Halpin, 2006). This method was also called Natural Language Information Analysis (NIAM) in earlier years (1970s-1980s). ORM includes procedures for mapping between conceptual and logical levels and is often used for data modeling during software engineering. The focus of ORM is on data modeling.

Because I will extend the existing tool, I first have created an ORM diagram of the XML schema that was designed for the original tool. Note that the ORM diagram serves as a conceptual schema. The actual XML Schema has been "flattened" because of XML's tree-like structure. The ORM diagram thus does not represent a "true" representation of how things are captured in XML but represents what is captured at a conceptual level. I will present the reader with an extended version of this ORM diagram in the Design section. Figure 5.8 shows the ORM diagram for the original GuideaMaps tool.

This ORM diagram provides a readable form of the structure of a GuideaTemplate. The ORM diagram shows that:

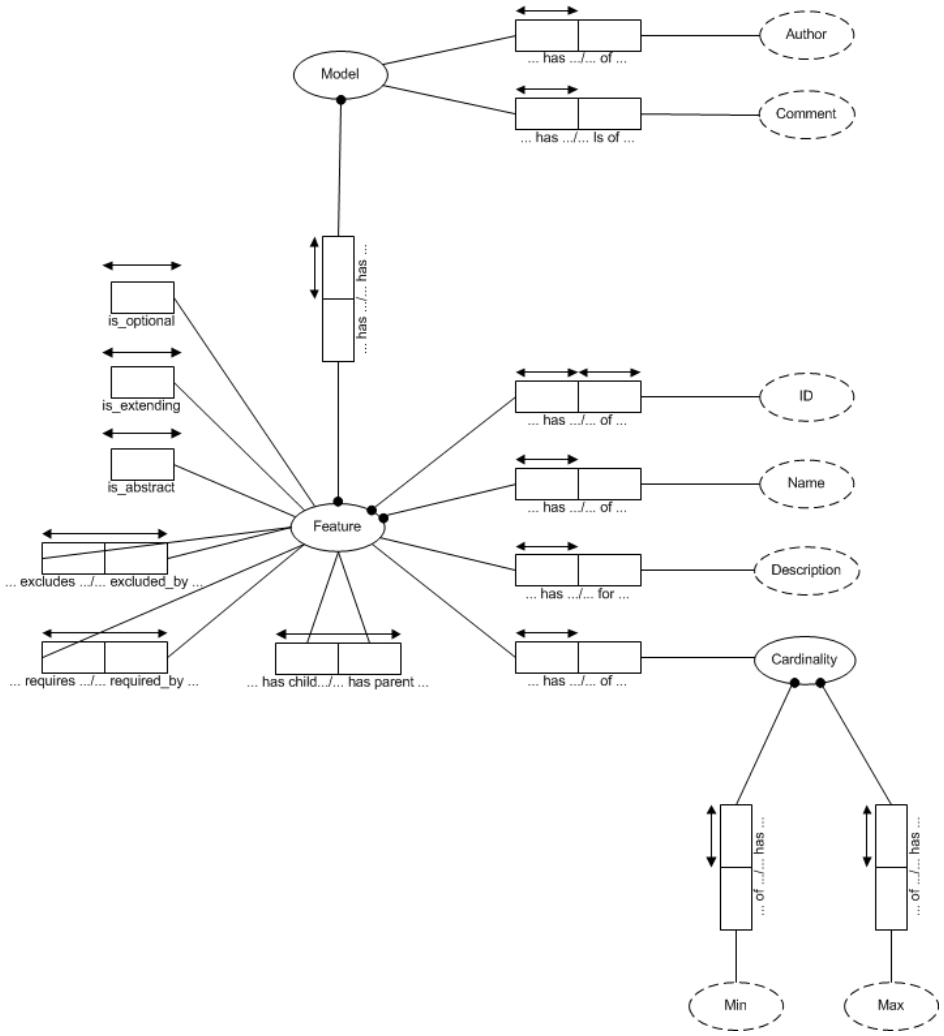

Figure 5.8: ORM diagram of the GuideaTemplate xml schema

A model contains:

- The name of the author of the model.
- A comment which is a textual description of the model.
- At least one feature, this will be the root feature of the model.

A feature contains:

- An ID that serves as a unique identifier.
- The name of the feature.

- A textual description of the feature.
- A cardinality which has a minimum and maximum value.
- A excludes or is excluded by relationship, a feature can exclude other features.
- A requires or is required by relationship, a feature can require other features.
- A child/parent relationship, a feature can have a child or parent feature.
- Can be optional, can extend or can be abstract.

When a user created a new GuideaMap, the GuideaMaps tool will read in the XML file containing the template and create a list of features based on the features described in the template file. Next, the tool will create guideas from this list of features and add information such as the child/parent relationship to these guideas. Finally, the tool will present a visual GuideaMap containing these guideas to the user. Figure 5.9 shows the end result that is presented to the user.

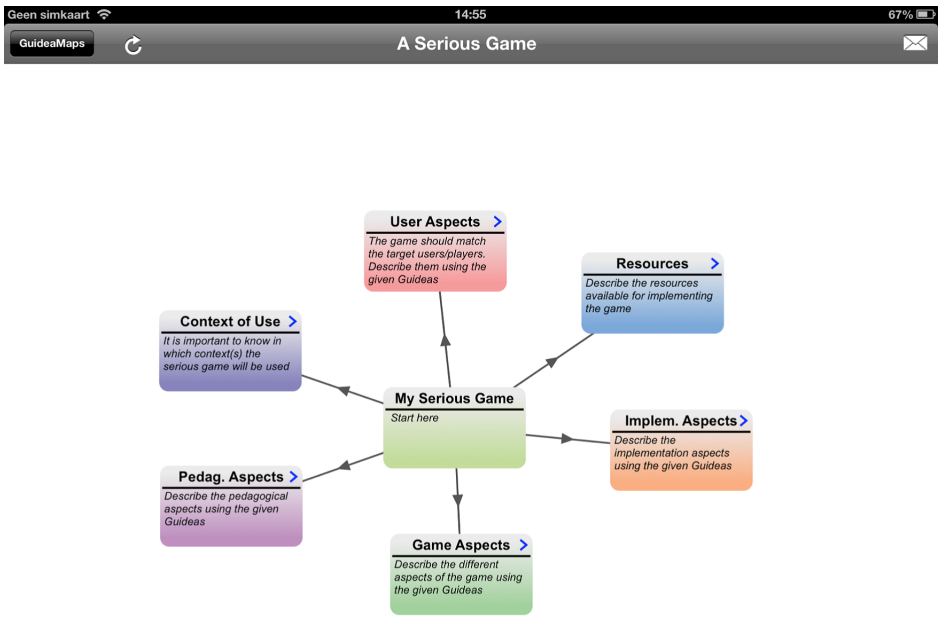

Figure 5.9: Example of a GuideaMap

5.3 Design

To integrate a dialogue mapping technique into the GuideaMaps tool, the IBIS method will be used. IBIS will be used as the graphical language to capture the discussion. As stated in the background section, IBIS consists of three different modeling elements and uses three grammatical rules. By using these three modeling elements and grammatical rules, it is possible to capture the discussion about any possible issue. The different elements can be connected and create a parent/child relationship, the grammatical rules put a constraint on this parent/child relationship.

Recall that the IBIS notation consists of the following three elements:

1. **Issues**: these are the issues that are being discussed by the decision-making group. Typically they are phrased as questions along the lines of "What will we do about x?", where x is the issue being discussed.

2. **Ideas**: ideas are proposed positions or solutions to the issues being discussed.

3. **Arguments**: Arguments can be used in defense (pro) or against (con) a certain position.

The IBIS grammar consists of the following three simple rules.

1. An issue can arise from an existing issue or idea. An issue can also arise from an argument. This means that an issue can arise from any other element in the IBIS notation

2. An idea can only arise from an existing issue.

3. A pro or con argument can only arise from an existing idea.

In the original GuideaMaps tool, a guidea is the equivalent of an issue that can be discussed by the decision-making group. A guidea will treat a requirement for the game to be developed during the requirement analysis phase. This could for example be the question: "What platform do we develop for?" To capture the full discussion process, every guidea is given an IBIS map instead of just a comment (as done in the original implementation). This IBIS map will contain the information about the discussion concerning a particular guidea.

In the IBIS notation, an issue element is an element that represents an issue that can be discussed by the decision-making group. This means that a guidea can serve as a root element of an IBIS map. Therefore, the guidea will

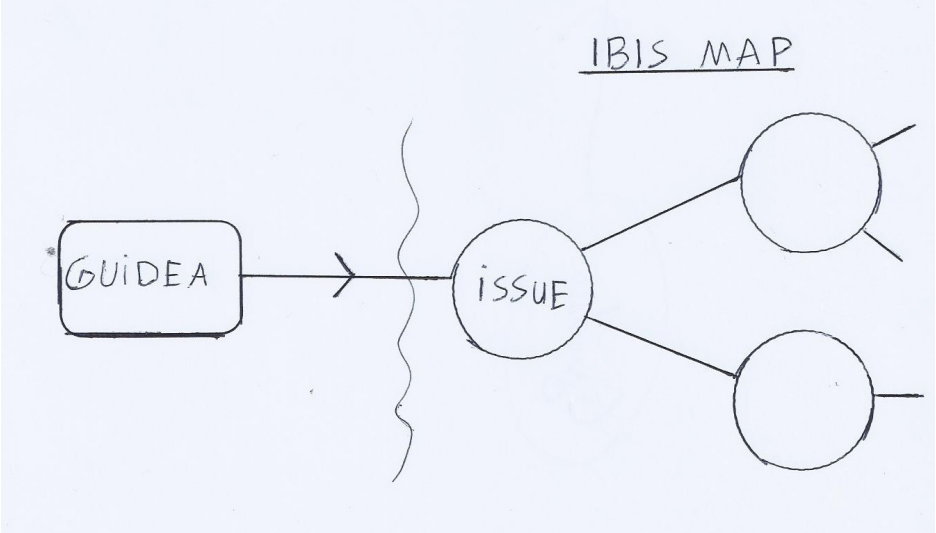

Figure 5.10: Sketch of the extended GuideaMaps tool

be converted into an issue element and will be used as the root element from where the IBIS map will start. A user can create a full IBIS map starting from this root issue element as shown in Figure 5.10.

The elements of the IBIS map will be called nodes. A node can contain additional information such as the creation date and time of a node, a comment or a description. Nodes can still be extended with additional information to introduce more context information into the IBIS map. In this thesis, I capture dates and times of creation in the background. Those can be used to distill information on the order requirements have been discussed, amongst others. These kinds of analyses, however, are outside of the scope of this thesis and could be investigated in the future (see Chapter 7).

5.3.1 Design Patterns

Some design patterns have been used in the development of the extended GuideaMaps tool. A design pattern is defined as follows:

"*In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.*"

[1]

Delegation

Delegation is a design pattern in object oriented programming where an object expresses certain behaviour to the outside but in reality delegates these tasks to a helper object. The helper object, which is known as the delegate, is given the responsibility to execute a task for the delegator.

An example is the window object in figure 5.11. The user clicks the close windows button but the window sends the "windowShouldClose" -message to its delegate. This gives the delegate the choice to close the window or not.

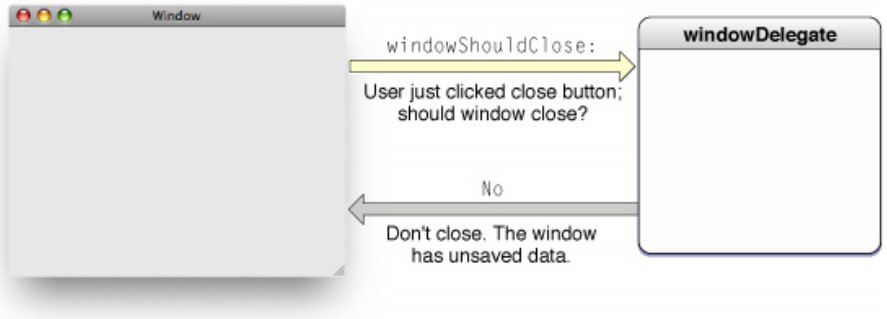

Figure 5.11: An example of the delegation design pattern (*iOS Delegation design pattern*, n.d.)

Model-View-Controller

The Model-View-Controller or MVC design pattern is used to develop the tool. Figure 5.12 shows an overview of the MVC design pattern. MVC is a software design pattern that divides the software application in three interconnected parts.

- The model component will directly manage the data, logic, and rules of the application. The model will respond to requests for information about its state and respond to instructions to change its state.
- The view component is a visual representation of information. This could be a chart, diagram or a web page. Multiple views can represent

[1] `https://en.wikipedia.org/wiki/Software_design_pattern`

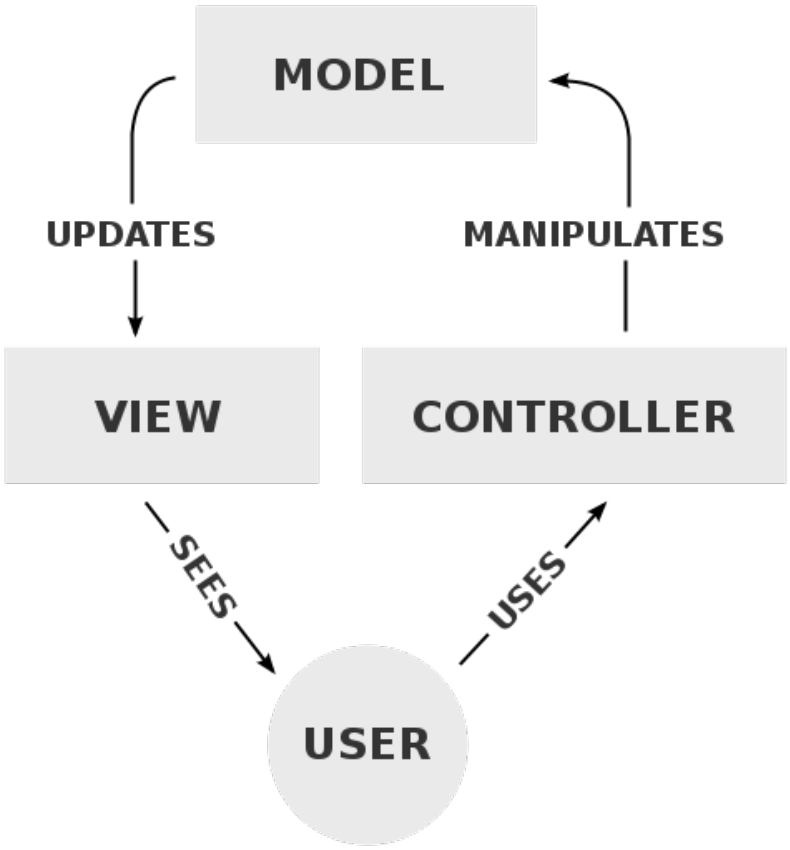

Figure 5.12: The Model-View-Controller design pattern (*MVC Design pattern*, n.d.)

the same information. The view will manage the display of information to the user.

- The controller component will accept input commands and translate them to the model or view component. A controller is the link between the user and the system and will inform the model and/or view to change its state.

The implementation of the extended GuideaMaps tool makes use of the MVC design pattern. The MVC design pattern is a 3-tier architecture. The application is separated in 3 different layers (the model, view and controller layers) and these different layers are separated. This defines how the objects communicate with each other. I will continue this section by explaining the design of these three layers.

5.3.2 The Model Layer

The representation of a node in the application will be split into two parts:

- A decision model: the decision model will represents information such as a node name, a description, a date and time of creation and a node count. It will also contain the information about what type of node this is.
- An application model: the application model will contain information that is relevant to the application. It will contain the x and y coordinates of a node on the screen, the information on the color of a node, whether it is expanded and enabled or not.

A node will be represented by two different classes. The MetaNode class contains the information relevant for the decision model and the Node class contains the information relevant to the application model. This is shown in Figure 5.13.

Together, these classes will represent the elements out of which an IBIS map can be created. An enumeration is used to define the different type of nodes.

Both of these classes can have a reference to a parent object and several child objects. This way, the parent/child relationship between the different elements in the IBIS map can be created. A node can only have one parent node but can have several child nodes.

The model layer also contains the NodeDocument, NodeModel and MetaModel classes as shown in Figure 5.14.

A NodeModel object contains a reference to a root Node object of an IBIS map. This NodeModel will contain information about all the nodes in the in memory IBIS map. It will also contain a reference to a MetaModel object which contains additional information about the application model such as the filename or the author name. This MetaModel object contains a reference to a root MetaNode object. The NodeModel object will be responsible for translating the current in memory model into the XML language.

Finally, the NodeDocument class extends the iOS UIDocument interface and will be responsible for saving and loading files into memory.

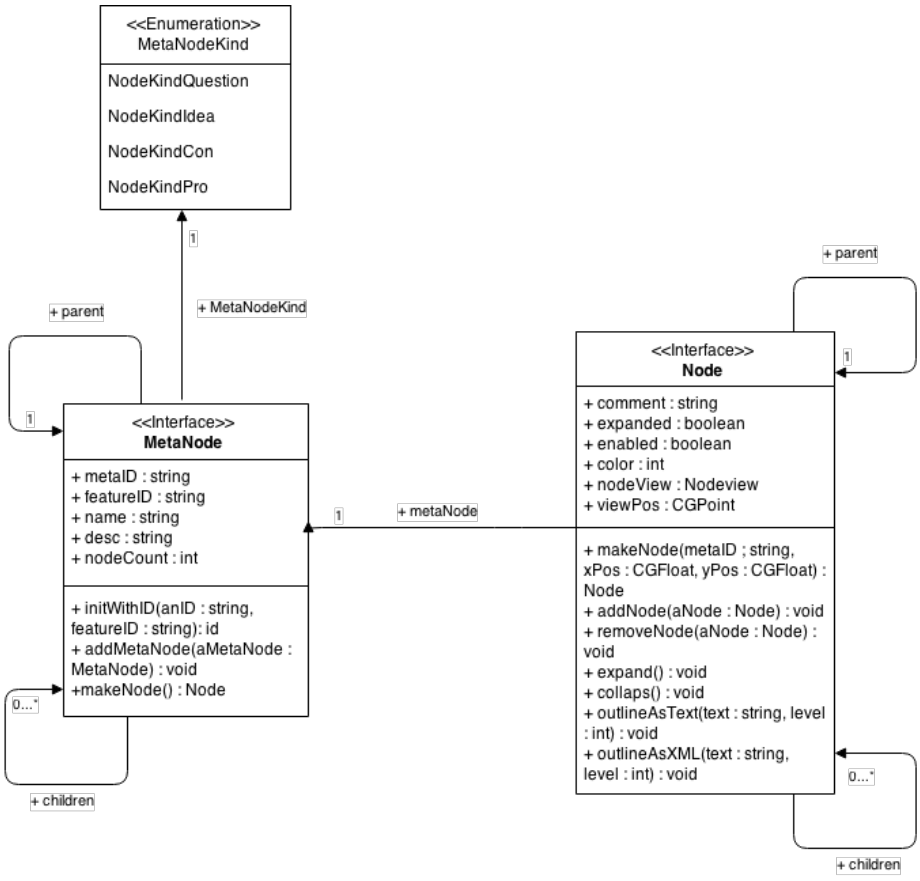

Figure 5.13: The representation of a node in the model layer

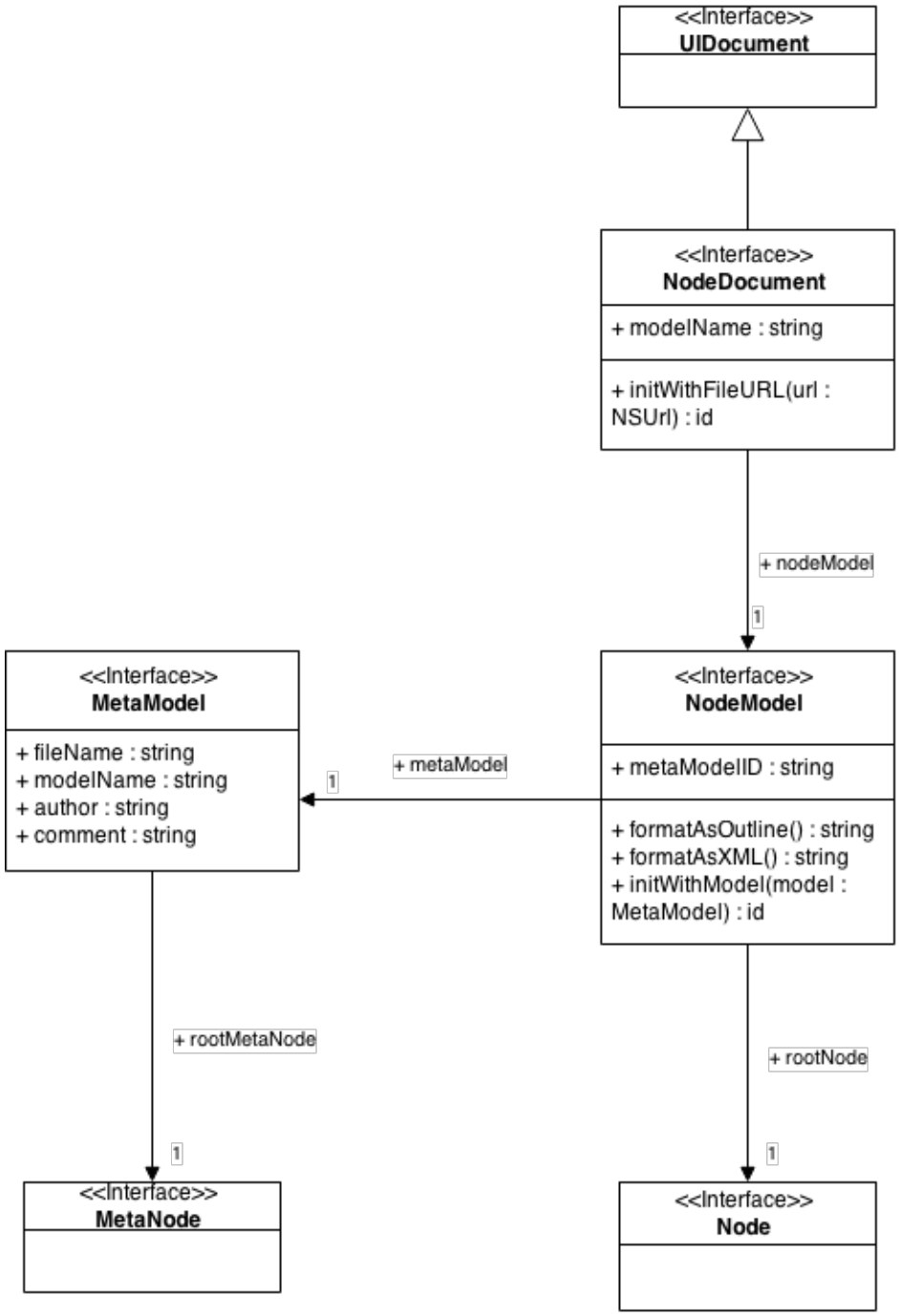

Figure 5.14: The extended GuideaMaps model layer

5.3.3 The View Layer

The view layer will contain everything concerning the visual representation of an IBIS map to the user. To implement the IBIS method into GuideaMaps, the application needs to be able to represent the different elements of which an IBIS map can consist of.

The NodeView class contains the logic for creating a visual representation of a Node object. It also contains the logic to select or deselect a view. It contains a reference to a Node object, a NodeName object and a NodeConnector object. Figure 5.15 shows the possible visual representations of Node objects depending on the type of node.

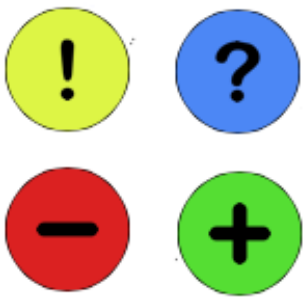

Figure 5.15: The visual representation of a node

4 types of nodes exist:

- A blue node with a question mark is the representation for an IBIS issue element.
- A yellow node with an exclamation mark is the representation for an IBIS idea element.
- A red node with a minus sign is the representation for an IBIS con argument element.

- A green node with a plus sign is the representation for an IBIS pro argument element.

The NodeName class contains the logic to draw text under a node. A user can create a node and add a name to this node. This name will serve as the description of a node as shown in Figure 5.16. A user can change this name at any time.

Figure 5.16: A GuideaMaps node with a name

The NodeConnector class contains the logic to draw an arrow line between two nodes. These lines are used to visually represent the parent/child relationship.

The NodeView class contains the logic to update the position of the name and connectors between a node when the user drags a node around on the screen or when a user zooms in or out of an IBIS map.

The final result will be an IBIS map where different elements are connected by their parent/child relationship and information can be added to a node in the form of a node name as show in Figure 5.17.

The NodePanel class encapsulates all the NodeViews that make up the active IBIS map and provides the logic necessary to handle user gestures such as panning, tapping and pinching. A NodePanel object contains a reference to a root NodeView and a selected NodeView. It also contains a reference to the action buttons (add, delete, collapse, expand). Figure 5.18 shows the full structure of the view layer.

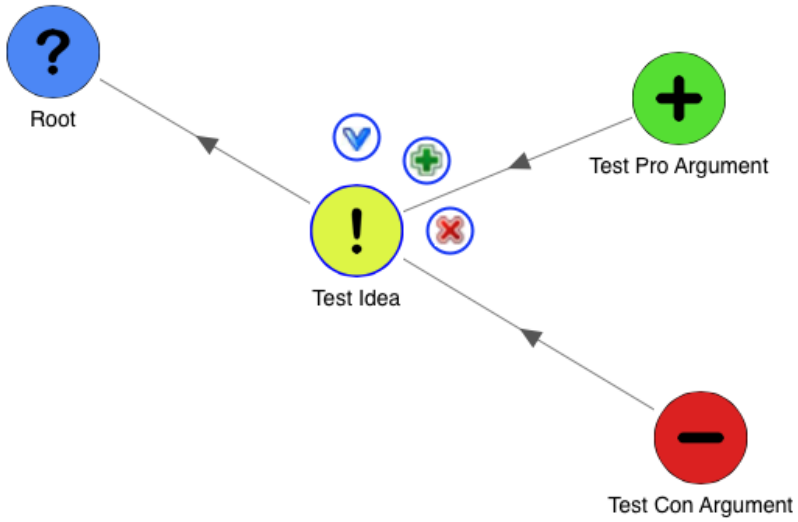

Figure 5.17: Example of an IBIS map

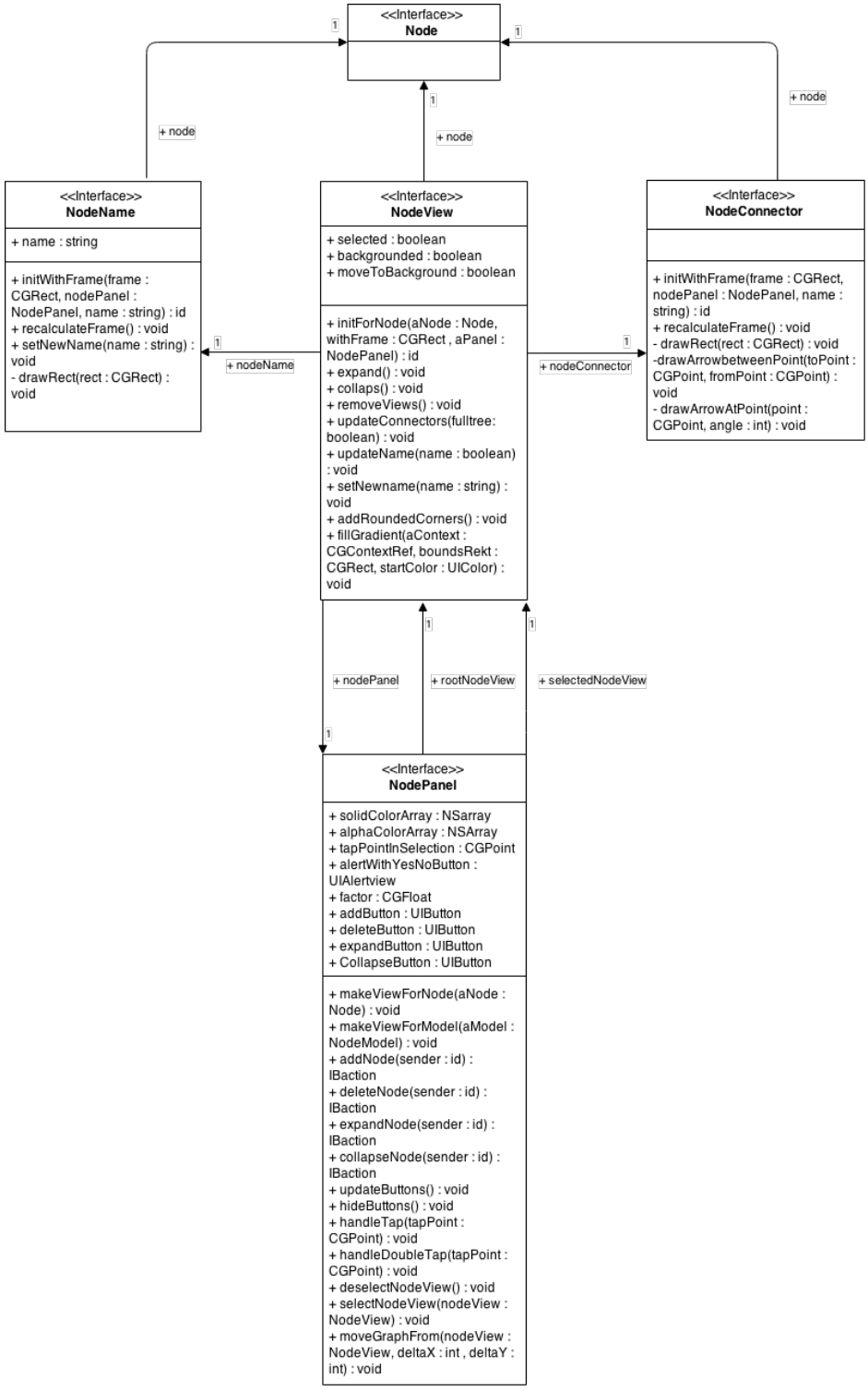

Figure 5.18: The extended GuideaMaps view layer

5.3.4 The Controller Layer

The controller is where everything comes together.

This controller implements the UIViewController interface. The UIViewController class provides the fundamental view-management model for all iOS apps. It is not advised to instantiate UIViewController objects directly. Instead, it is advised to instantiate subclasses of the UIViewController class based on the specific task each subclass performs.

A view controller:

- resizes and lays out its views
- adjusts the contents of the views
- acts on behalf of the views when the user interacts with them

The IBISMapController will be responsible for the coordination between the model that is kept in the NodeDocument and the visualization in the NodePanel. It is also responsible for checking if an IBIS map exists when a user double tabs on a Guidea. If no IBIS map exists for a particular Guidea, the controller will create a new IBIS map. If an IBIS map for this particular Guidea already exists, the controller will load this map into memory. To be able to create a new IBIS map or load an existing IBIS map, the controller has a reference to the NodeService and MetaService classes.

It also contains a reference to a Guidea, this is the Guidea for which the IBIS map will be created. The controller layer is shown in Figure 5.19.

Other view controllers that are being used in the extended application are:

- NodeDetailViewController: popover controller used to edit node details.
- NodeTypeNavController: popover controller used to add new nodes.
- NodeTypeTableController: table controller used to show a list of possible nodes to add.
- NodeTypeDetailsController: table controller for type details.

The node type controllers are used when a user tries to create a new node. When clicking the add node button, a popover controller will show up with the possible nodes a user can add. The possible nodes that a user can add as a child node to an existing node will depend on the grammatical rules of the IBIS method. This is shown in Figure 5.20.

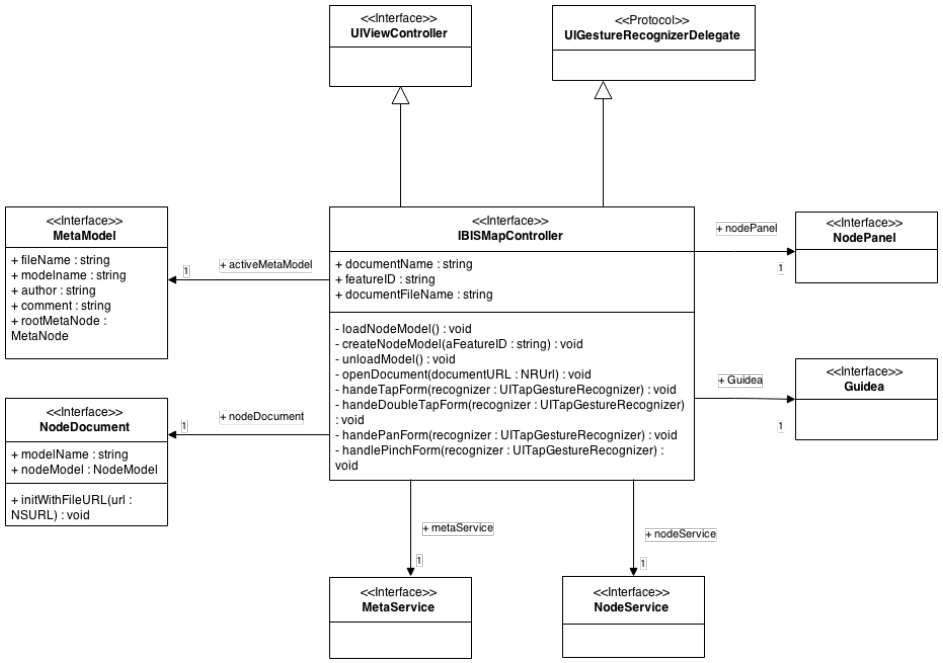

Figure 5.19: The extended GuideaMaps controller layer

5.3.5 Services and Parsers

The services and parser classes are used to read the content from an XML file and transform them into an application model. The NodeParser class will parse an XML file and create a node model with a root node. The MetaNodeParser will create a meta model with a root meta node.

The service classes work together with the parser classes to maintain the application model in memory. When opening an IBIS map, the service classes will make sure the content is read from the right XML file and the application model is created. These service classes are also used to write an IBIS map that is in memory to an XML file on the iPad device. Every new GuideaMap will create a folder with the name of the GuideaMap. In this folder, a folder with the name models will be created by the service classes. This is the folder where the IBIS map application models will be saved as XML files. The service and parser classes are shown in Figure 5.21.

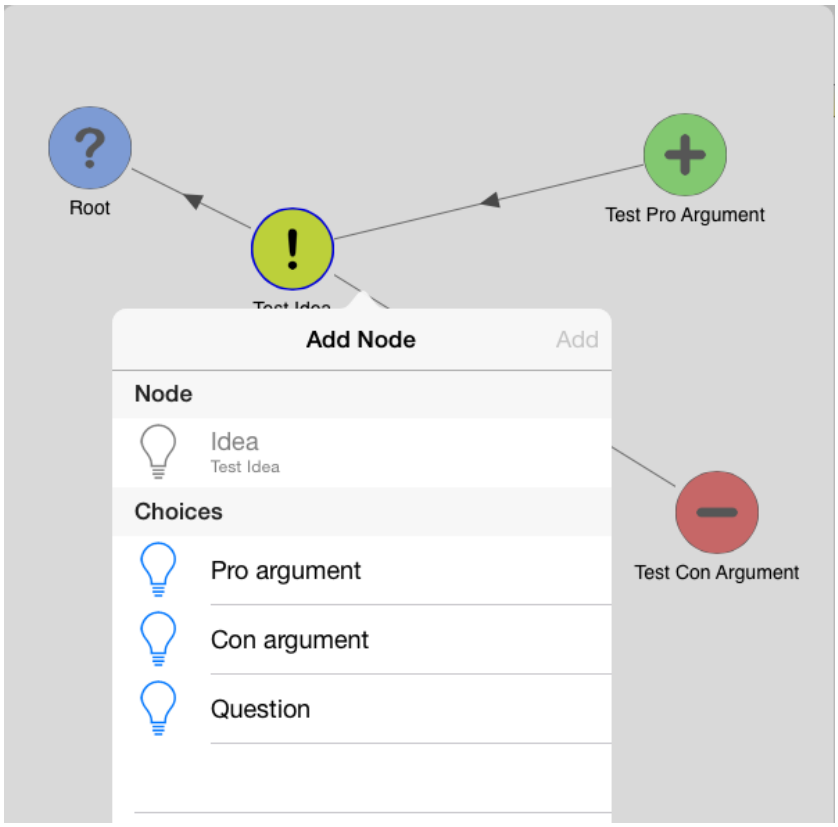

Figure 5.20: Adding a node to an IBIS map

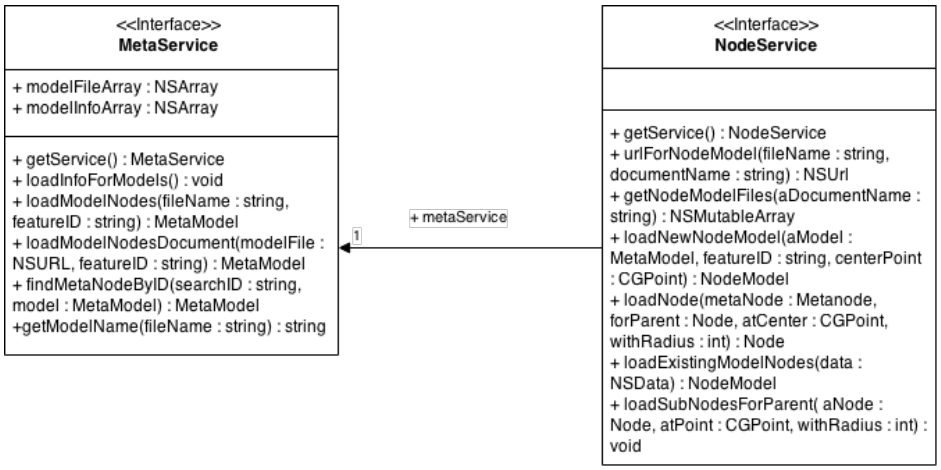

Figure 5.21: The MetaService and NodeService classes

5.4 Implementation

5.4.1 Introduction

In this section, the implementation of the extended GuideaMaps tool is discussed in more detail. To be more precise, the creation and visualization of an IBIS map and the reading and writing of documents will be discussed.

5.4.2 File Structure

The application has been developed using the MVC pattern. Figure 5.22 shows an overview of the file structure of the application. The files are split up into the "model", "view and "controller" folders according to the MVC pattern. The "Services" and "Parsers" folders contain the classes that are used to read and write documents.

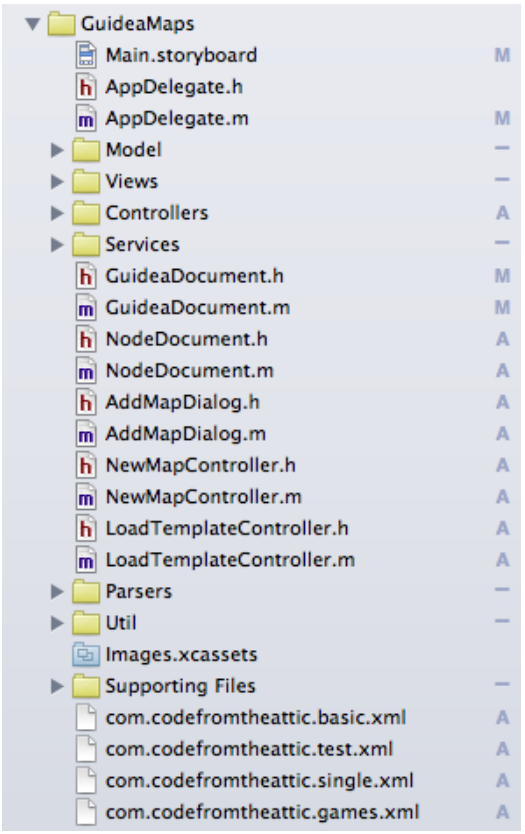

Figure 5.22: The file structure of the extended GuideaMaps tool

5.4.3 The Controller Layer

The IBISMapController class will check if an IBIS map exists or not for a particular Guidea. If no existing file was found, a new one will be created. If an existing file was found, this file will be opened. The files are stored as XML documents in the iPad memory. This code is shown in Figure 5.23.

```
//load model into memory or create new one
- (void) loadNodeModel {
    self.documentFileNames = [self.nodeService getNodeModelFiles:self.documentName];

    NSString *fullFileName = [NSString stringWithFormat:@"%@.xml", self.featureID];

    if ([self.documentFileNames count] > 0 && [self.documentFileNames containsObject:fullFileName]) {
        // load ibis map
        NSURL *docURL = [self.nodeService urlForNodeModel:fullFileName documentName:self.documentName];
        [self openDocument:docURL];

    } else {
        //create new ibis map
        [self createNodeModel:self.featureID];
    }
}
```

Figure 5.23: The method used to load a node model

If no existing IBIS map was found in memory, a new one will be created. Figure 5.24 shows the process of creating a new IBIS map.

This process consists of the following steps:

- A new NodeDocument object is created.
- A MetaModel object is created with a root MetaNode. This root MetaNode will be the Guidea that is transformed into an IBIS issue element.
- A NodeModel object is created and this is added to the NodeDocument object.

If an existing IBIS map was found in memory, it will be loaded. The NodeParser class will parse the XML document and create a NodeDocument containing a NodeModel. The MetaNodeParser class will create a MetaModel and this MetaModel will also be added to the NodeDocument. This code is shown in Figure 5.25.

```
- (void) createNodeModel:(NSString *)aFeatureID  {

    if (aFeatureID == nil) {
        NSLog(@"ModelName is empty");
        return;
    }

    // save and remove old model
    if (self.activeDocument != nil) {
        [self.activeDocument savePresentedItemChangesWithCompletionHandler:nil];
        [self unloadModel];
        self.activeDocument = nil;
    }

    NSString *fullFileName = [NSString stringWithFormat:@"%@.xml", aFeatureID];
    NSURL *saveUrl = [self.nodeService urlForNodeModel:fullFileName documentName:self.documentName];
    self.activeDocument = [[NodeDocument alloc] initWithFileURL:saveUrl];
    self.activeDocument.modelName = aFeatureID;

    self.activeMetaModel = [[MetaModel alloc] init];
    MetaNode *metaNode = [[MetaNode alloc] initWithID:@"rootNode" featureid:aFeatureID];
    metaNode.desc = self.guidea.metaGuidea.name;
    metaNode.name = self.guidea.metaGuidea.name;
    metaNode.kind = NodeKindQuestion;
    self.activeMetaModel.rootMetaNode = metaNode;

    NodeModel *model = [self.nodeService loadNewNodeModel:self.activeMetaModel featureID:aFeatureID
        centerPoint:CGPointMake(20, 20)];

    self.activeDocument.nodeModel = model;
    [self.activeDocument updateChangeCount:UIDocumentChangeDone];

    [self.nodePanel makeViewsForModel:model];
    self.nodePanel.rootNodeView = model.rootNode.nodeView;

    [self.nodePanel setNeedsDisplay];
}
```

Figure 5.24: The method used to create a new node model

```
- (void)openDocument:(NSURL *)documentURL
{
    // check if reopening the existing document
    if ([self.activeDocument.fileURL isEqual:documentURL])
        return;

    if (self.activeDocument != nil) {
        [self unloadModel];
        [self.activeDocument savePresentedItemChangesWithCompletionHandler:nil];
    }

    // Open document from given URL
    self.activeDocument = [[NodeDocument alloc] initWithFileURL:documentURL];
    [self.activeDocument openWithCompletionHandler:^(BOOL success) {
        if (success) {
            // Enrich model and update the view
            if (self.activeDocument != nil && self.activeDocument.nodeModel != nil) {
                if (self.activeDocument.nodeModel.metaModelId == nil) {
                    // metaModelName was not found in file
                    NSLog(@"metaModelName not found!");
                }
                if (self.activeDocument.nodeModel.rootNode == nil) {
                    // no nodes found in file
                    NSLog(@"No nodes found!");
                }

                self.activeDocument.modelName = self.featureID;
                self.activeMetaModel = [self.metaService loadModelNodesDocument:documentURL
                    documentName:self.documentName featureId:self.featureID];
                if (self.activeMetaModel.rootMetaNode == nil) {
                    NSString *msg = [NSString stringWithFormat:@"The NodeTemplate with ID '%@' couldn't
                        be found!", self.featureID];
                    UIAlertView *alert =
                    [[UIAlertView alloc] initWithTitle:@"Error Loading GuideaTemplate"
                                               message:msg
                                              delegate:nil
                                     cancelButtonTitle:@"OK"
                                     otherButtonTitles: nil];
                    alert.alertViewStyle = UIAlertViewStyleDefault;
                    [alert show];
                    self.activeDocument = nil;
                    return;
                }
                self.activeDocument.nodeModel.metaModel = self.activeMetaModel;

                // enrich nodeModel with proper metaNodes
                [self.nodeService enrichModelNodes:self.activeDocument.nodeModel
                                       inMetaModel:self.activeDocument.nodeModel.metaModel];
                // instantiate views in guideaPanel for guideaModel
                [self.activeDocument updateChangeCount:UIDocumentChangeDone];
                [self.nodePanel makeViewsForModel:self.activeDocument.nodeModel];
                self.nodePanel.rootNodeView = self.activeDocument.nodeModel.rootNode.nodeView;
            } else
                self.nodePanel.rootNodeView = nil;

            [self.nodePanel setNeedsDisplay];
        } else {
            NSLog(@"failed to open!");
        } }];
}
```

Figure 5.25: The method used to open an existing node model

5.4.4 Reading and Writing Documents

The application will save created IBIS maps in the form of an XML document and load these XML documents back into memory by parsing them and transforming them into an IBIS map.

An instance of a NodeDocument class will contain a nodeModel, which

contains the information of an IBIS map, and will be responsible for reading and writing IBIS maps. Figure 5.26 and 5.27 show the reading and writing methods in the NodeDocument class. The NodeDocument class implements the UIDocument interface which is used to read and write documents.

```
// writing document
- (id)contentsForType:(NSString *)typeName error:(NSError **)outError {
    NSLog(@"writing document for URL %@", self.fileURL);

    NSString *text;
    if (self.nodeModel.rootNode != nil) {
        text = [self.nodeModel formatAsXML];
    } else {
        text = @"Empty file";
    }
    NSData *docData = [text dataUsingEncoding:NSUTF8StringEncoding allowLossyConversion:NO];
    return docData;
}
```

Figure 5.26: The method used for writing a document

The nodeModel will call the formatAsXML method to transform the model into XML data and the XML document will be saved.

```
// reading document
- (BOOL)loadFromContents:(id)contents ofType:(NSString *)typeName error:(NSError **)outError {
    NSLog(@"reading document for URL %@", self.fileURL);

    if ([contents length] == 0)
        return FALSE;

    NodeService *service = [NodeService getService];
    self.nodeModel = [service loadExistingModelNodes:contents];

    if (!self.nodeModel.rootNode  || !self.nodeModel.rootNode) {
        NSLog(@"Error loading nodeMap for URL %@", self.fileURL);
        return FALSE;
    }
    return TRUE;
}
```

Figure 5.27: The method used for reading a document

To read an XML file and transform it into an IBIS map. The NodeService class will be used. This class will call the parser which will parse the XML document and return a NodeModel. This is shown in Figure 5.28.

```
-(NodeModel *) loadExistingModelNodes: (NSData *) contents {
    NodeParser *parser = [[NodeParser alloc] init];
    NodeModel *nodeModel  = [parser parseDocumentWithData:contents];
    return nodeModel;
}
```

Figure 5.28: The method used to load an existing node model

Figure 5.29 shows an example of an IBIS map converted into an XML file. The XML is stored on the iPad device and contains the information needed

to load an IBIS map. It contains the different nodes and some information about them such as: the node type, the color of a node and the x and y coordinates to determine the position of a node on the screen. The XML file will also contain the ID of the Guidea for which this IBIS map has been created.

```
<nodeModel metaModel="test”>
  <feature id="testFeature”>
    <node id="rootNode" xPos="34" yPos="73" color="0" type="0" expanded="TRUE”>
      <nodename>Question</nodename>
      <nodedescription>This is a test issue</nodedescription>
        <node id="QqgfDZ5SxX" xPos="216" yPos="196" color="1" type="1" expanded="TRUE”>
          <nodename>Idea</nodename>
          <nodedescription>This is a test idea</nodedescription>
            <node id="X8mKJrlswD" xPos="486" yPos="336" color="2" type="2”>
              <nodename>Con argument</nodename>
              <nodedescription>This is a test con argument</nodedescription>
            </node>
            <node id="iHCds9sJgP" xPos="463" yPos="78" color="3" type="3”>
              <nodename>Pro argument</nodename>
              <nodedescription>This is a test pro argument</nodedescription>
            </node>
        </node>
    </node>
  </feature>
</nodeModel>
```

Figure 5.29: Example of an XML file created from an IBIS map

The parser class is used to convert this XML document into a model that represents the IBIS map in the application. The parser classes implement the NSXMLParserDelegate to read and write XML Documents. This is shown in Figure 5.30.

5.5 Summary

In this chapter, I have presented the analysis of the original GuideaMaps tool and the prerequisites needed to develop the extended tool. The prerequisites consisted of the iOS platform, the Xcode integrated development environment and the Objective-C programming language.

I have also presented the design of the extended tool. The overall structure of the application and the functionalities are presented.

Finally, the implementation of the extended GuideaMaps was presented. This section takes a more in-depth look into the creation and visualization of an IBIS map and the reading and writing of documents.

```
-(NodeModel *)parseDocumentWithData:(NSData *)data {
    if (data == nil)
        return nil;

    // this is the parsing machine
    nodeModel = [[NodeModel alloc]initWithModel:nil];
    NSXMLParser *xmlparser = [[NSXMLParser alloc] initWithData:data];

    // this class will handle the events
    [xmlparser setDelegate:self];
    [xmlparser setShouldResolveExternalEntities:NO];

    // now parse the document
    BOOL ok = [xmlparser parse];
    if (ok == NO)
        NSLog(@"Error while parsing document");
    nodeModel.rootNode = rootNode;
    return nodeModel;
}
```

Figure 5.30: The method used to parse an XML document

6 Evaluation and Results

In this chapter, the evaluation and its result will be presented. The evaluation is done by means of a case study consisting of a use-case scenario and an interview using a questionnaire. The setup, the methodology used for the evaluation and the results will be discussed.

6.1 Setup

The extended GuideaMaps tool was evaluated by letting 7 groups of at least 3 people use the tool in a use case scenario. During these use case scenarios, a GuideaMaps template for making decisions about serious games was used. The participants were asked to decide on who the main character of their game would be. In each group, one person was chosen to perform the role of the dialogue mapper. The dialogue mapper was the person using the tool, while the other participants followed the discussion on a shared display using a projector.

Before the participants used the tool, they were given information on the background and goals of the experiment. They were also given a short introduction to the tool and the IBIS framework. This was done by means of a presentation (included in Appendix B). The participants were only given a basic introduction to the tool because the ease of use and the learnability were two factors that I wanted to evaluate.

The different groups consisted of participants with a different technical background. This mix of people with a different technical background was made on purpose in order to evaluate whether there was a difference in usage and learnability of the tool between people with a different technical background. The participants with a technical background consisted out of people working in the software industry or a related technical industry. The participants without a technical background had different backgrounds, ranging from people who work as a teacher in middle school to people who work in the financial industry. The age of the different participants was mixed, ranging from the age of 18 to 55.

The evaluation of each group consisted of two phases. Two questionnaires were used to evaluate the tool. The SUS or Systems Usability Scale questionnaire (Brooke, 1996) was used to measure the usability and learnability of the tool. The other questionnaire consisted out of some open questions that were used to evaluate the opinions of the participants about different aspects of the tool.

6.2 Methodology

Each evaluation consisted of two phases, each of these phases having their respective evaluation method. In the first phase, the general technical knowledge of the participants was collected. During the second phase, the tool was used and afterwards evaluated by the participants. The following sections describe the different methods used for the evaluation of the extended GuideaMaps tool.

6.2.1 Pre-experiment

Before the experiment, the participants were asked questions about their technical knowledge. The participants were asked how many times per week they use computer systems, if they had experience with touch screen interfaces and if they had experience with other tools that could be used to support meetings or decision-making.

6.2.2 Experiment

First, the participants were given an introduction by means of a PowerPoint presentation. The presentation informed the participants on the setup, background and goals of the experiment and introduced them to the extended GuideaMaps tool. This presentation had duration of ten minutes.

The participant who took the role of the dialogue mapper was also given a short introduction to the IBIS framework. He was informed about the three different elements that are used in the IBIS framework, namely "Issues", "Ideas" and "Arguments" and the three rules that can be used to connect these elements. A short introduction to the tool was given because the ease of use, intuitiveness and learnability of the tool are aspects to be evaluated. Participants with a lot of technical knowledge and experience were given the same introduction as people with less experience with computer systems. This introduction only lasted for 2 to 3 minutes.

The Dialogue Mapper's Evaluation

The dialogue mapper's evaluation of the tool was collected by means of a questionnaire. The SUS (Systems Usability Scale) questionnaire was used for this. This is a questionnaire that is used to measure the ease of use, usability and learnability of software and it can be used on a small sample size while still being reliable and valid (Brooke, 2013). The SUS questionnaire was filled in after the tool was used. The idea was to record the immediate impression of the user without giving him much time to think about each statement. No discussions about the tool were allowed before filling in the questionnaire. The SUS is a 10 item questionnaire with an inverted Likert scale from 1 (Strongly disagree) to 5(Strongly agree).

The following questions were used:

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use
- I felt very confident using the system.

- I needed to learn a lot of things before I could get going with this system.

The SUS uses the following response format:

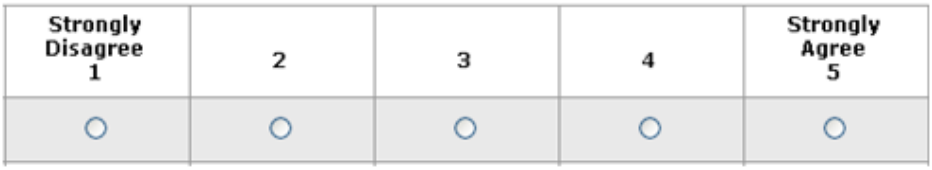

Figure 6.1: The SUS questionnaire response format

The Other Participants' Opinion

The participants who were also part of the experiment but did not take on the role of the dialogue mapper were also questioned by means of a questionnaire. This questionnaire consisted of 6 questions that were used to get to know the opinion of the participants about the tool. These questions could be answered with yes or no but the participants also had the option to add additional comments to the answers.

The following questions were used:

- Was the organization of decisions on the screen clear?
- Was the meaning of the different colors clear?
- Do you think the discussion was easy to follow on the screen?
- Is there something you did not like about the platform, and would like to change?
- Do you think this software provides added benefits to regular requirement analysis/meetings?
- Would you want to use this software in the future?

6.3 Results

In this section, the results of the evaluation will be given.

6.3.1 Pre-experiment

All participants from the group with a technical background as well as the participants from the group without a technical background reported that they use computer devices on a daily base. All of the participants also reported having experience with mobile devices using a touch screen interface such as smartphones or tablets. Only two of the participants had used software before that supports a meeting process. Both of these participants were from the group without a technical background and both of them reported to have used a mind mapping tool for this.

6.3.2 Experiment

The SUS Questionnaire

The SUS questionnaire was filled in by a total of eight participants. Four of these participants had a technical background and the other four had no technical background. The results are split up according to these two groups.

Table 6.1 shows the results of the questionnaire filled in by the group of the participants with a technical background:

Table 6.1: SUS scores of the participants with a technical background

User	1	2	3	4
Question 1	1	4	3	3
Question 2	2	2	2	2
Question 3	4	4	4	4
Question 4	1	1	1	1
Question 5	4	4	4	5
Question 6	2	3	1	1
Question 7	3	4	5	4
Question 8	3	2	1	2
Question 9	4	4	5	4
Question 10	1	2	1	1

Table 6.2 shows the results of the questionnaire filled in by the group of participants without a technical background:

Table 6.2: SUS scores of the participants without a technical background

User	1	2	3	4
Question 1	4	4	4	2
Question 2	1	1	1	1
Question 3	4	5	5	4
Question 4	1	1	1	2
Question 5	3	4	4	3
Question 6	2	2	1	1
Question 7	5	4	4	5
Question 8	2	2	2	1
Question 9	4	4	5	4
Question 10	1	2	1	2

To calculate the score for these results, the following rules are used:

- For odd numbered items (1, 3, 5, 7, 9), subtract one from the user response.
- For even numbered items (2, 4, 6, 8, 10), subtract the user response from 5.

By doing this, all the responses are scaled down to a 0 - 4 score, where 0 is the most negative and 4 is the most positive score.

Next, all the responses are added and the total result is multiplied by 2.5 to get the total score. This will convert the range of possible values from 0 to 100 instead of 0 to 40.

Table 6.3 shows the total scores for the group of participants with a technical background:

Table 6.3: Total scores of the participants with a technical background

User	1	2	3	4
Question 1	0	3	2	2
Question 2	3	3	3	3
Question 3	3	3	3	3
Question 4	4	4	4	4
Question 5	3	3	3	4
Question 6	2	3	4	4
Question 7	2	3	4	3
Question 8	2	3	4	3
Question 9	3	3	4	3
Question 10	4	3	4	4
Total score	67.5	75	87.5	82.5

Table 6.4 shows the total scores for the group of participants without a technical background:

Table 6.4: Total scores of the participants without a technical background

User	1	2	3	4
Question 1	3	3	3	1
Question 2	4	4	4	4
Question 3	3	4	4	3
Question 4	4	4	4	3
Question 5	2	3	3	2
Question 6	3	3	4	4
Question 7	4	3	3	4
Question 8	3	3	3	4
Question 9	3	3	4	3
Question 10	4	3	4	3
Total score	82.5	82.5	90	77.5

How to interpret the SUS questionnaire scores?

According to research done in a paper by Bangor A, Kortum P and Miller J (Bangor et al., 2009), the average SUS score is around 70. This research takes into account 3500 surveys within 273 studies over the last 10 years. The paper also proposes a scoring metric as shown in Figure 6.2. A score of 70 is seen as the average score, everything above 70 is considered above average and everything lower than 70 is considered lower than average.

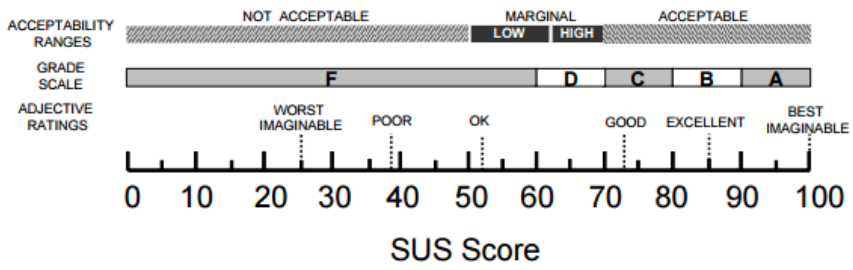

Figure 6.2: The SUS questionnaire scoring metric (Bangor et al., 2009, p. 121)

Some findings that can be derived from the participants SUS scores are:

- Seven out of eight participants have scored the tool as above average.

- One out of eight has scored the tool as below average.
- Six out of the eight scores are in-between the good and excellent rating.
- The scores in the group with participants without a technical background seem to be higher on average.
- The participant that scored the tool as below average is from the group with participants with technical knowledge.

Participants' Opinion

The following questions were answered with a yes or no by the participants not functioning as the dialogue mapper. They also had the option to add an additional comment to each question.

Was the organization of decisions on the screen clear?

Twelve out of fourteen participants answered "yes" to this question. Some participants did notice that the organization of the decisions on the screen could become too cluttered or overwhelming when the decision map started to grow. Some more structure in the decision map is needed when it becomes bigger. One participant also noted that the original issue element and the first idea elements should stand out more because when the map becomes bigger these elements that are important to the discussion can be overseen or disappear in the overwhelming structure.

Was the meaning of the different colors clear?

Thirteen out of fourteen participants answered "yes" to this question. From perceiving the usage of the tool by the participants, it was noted that one participant had problems with the red button that is used to delete an element from the map. He mistakenly used it as the button to add a con argument. This happened because a con argument and the button to delete an element both have the red color.

Do you think the discussion was easy to follow on the screen?

Thirteen out of fourteen participants answered "yes" to this question. A remark was made by several participants that the overview of the discussion could become less clear when the decision map starts growing, making the discussion less easy to follow.

Is there something you did not like about the platform, and would like to change?

The following remarks were made:

- The original issue should be clearly visible at all times.
- The screen can become cluttered when a decision map becomes too big.

- There is no distinction between strong and weak arguments.
- The dialogue mapper cannot get involved in the discussion because otherwise he can steer the process in his advantage.
- Introduce a fixed structure to the decision map. Right now a user can drag an element anywhere he wants on the map.

Do you think this software provides added benefits to regular requirement analysis/meetings?

Out of the fourteen participants, twelve of them thought the tool added benefits over regular meetings or decision-making. One of the participants who answered "no" did think there was potential if some changes were made to the tool. These changes would have to be in the organization of the decisions. Some participants reported that the visible overview of the whole discussion helped them in to think about the issue in a different way.

Would you want to use this software in the future?

Ten out of fourteen participants answered "yes" to this question. Opinions are mixed about this question. One participant who answered "no" did state that he would use the software if some changed to the organization and visualization of the elements on de decision map were made.

6.4 Discussion

According to the SUS questionnaire, seven out of eight participants have scored the extended GuideaMaps tool as above average. Most scores reside between the good and excellent scale. I can conclude that the tool and the IBIS framework are indeed easy to use and easy to learn for people with as well as people without a technical background. No extensive training is needed for a person to use the tool.

The participants who joined the discussion without using the tool themselves also stated that the discussion was easy to follow on the screen. I can conclude that these people also had no problem with following the discussion using the IBIS framework. But some problems have to be taken into consideration:

- The organization of the different elements in the decision map will become too overwhelming as the map grows. Several participants stated that the map became cluttered and the discussion way less easy to follow as the decision map grew.

- The original issue element has to stand out more because it will disappear into the background as the decision map grows. This also counts for the ideas that are brought up, the original ideas of a sub tree should stand out more so that focus on it is not lost.

- Arguments should be able to have a weight. Some arguments are more important than other arguments or can completely nullify other arguments. When an idea element has five pro arguments and one con argument, at first it seems like this is a good idea because it has many pro arguments. But the con argument can be so strong that the five pro arguments lose their strength.

- Users can place the elements on the decision map anywhere. Because of this they are free to structure the decision map in any way they like. But it might be better to introduce a fixed structure for the elements in order to keep a better overview of what is happening on the decision map and to fix the problem of cluttered information on the screen.

The role of the dialogue mapper is important. The dialogue mapper should take the role of a person that is in charge of a panel of speakers but should not be involved into the discussion himself. When a dialogue mapper gets involved in the discussion, it is easy for this person to steer the conversation into the direction he wants. As stated in the background section, a dialogue mapping technique was chosen because it filters out effects such as peer pressure or other social influences during meetings. But when a dialogue mapper gets involved into the discussion these factors can be introduced again. The dialogue mapper should also be proficient in using the IBIS framework and transforming a discussion into a decision map.

The opinions on the extended GuideaMaps tool being useful were mostly positive. Most people saw themselves using a tool like this in the future. One person did remark that some changes to the organization of the decision map should be made before he would use it again. Some people did not see the added benefits of using a tool like this and did not see them using this tool in the future. The reason for this was not given. I think that the reason is more general and related to resistance in using systems or processes that capture design rationale. Some people perceive this as a waste of time, while others are convinced that this could improve their productivity.

7 Conclusion

7.1 Introduction

This final chapter concludes this thesis by reflecting on the work done to develop the extended GuideaMaps tool. Limitations and future work are discussed as well.

7.2 Summary

The original GuideaMaps tool was developed to support the requirement analysis process of serious games and documents what decisions are made, but not why they are made. The goal of this thesis was to extend GuideaMaps with support for capturing the discussion and rationale behind decisions made. The aim was to make the decision making process more transparant and traceable, allowing the stakeholders to look back and understand how some decisions came to be. The authors of the original GuideaMaps paper stated that the tool should be usable and understandable by all stakeholders participating in the decision-making process and that some guidance was needed during this process.

To capture the rationale behind the decision-making process, a dialogue mapping technique was introduced into the GuideaMaps tool. The three

requirements for a dialogue mapping technique were:a shared display, a dialogue mapper and a graphical language (J. Conklin, 2005).

The iPad tablet device, for which the original tool was developed, connected to a beamer (or an external screen big enough for the stakeholders to observe) can serve as the shared display. One person would take on the role of the dialogue mapper using the actual tool while the other participants can see the overview of the discussion on the shared display. By displaying the overview of the full discussion at all times, problems is a meeting setting such as short-term memory or circular logic could be addressed.

The IBIS framework was chosen as the graphical language needed for dialogue mapping. The IBIS framework was chosen because it is a simple and intuitive language. No extensive training is needed to understand it. Other graphical languages have been considered, some of which have more expressive power than the IBIS language. But by adding more expressive power, in general more complexity is added. Since the tool should be usable by the different stakeholders participating in the decision-making process, i.e. people with a technical background as well as people without a technical background, the IBIS framework was chosen.

The extended GuideaMaps tool was evaluated by letting groups of people use it in a use case scenario. This evaluation showed that the tool was usable by people with as well as without a technical background and that no training was needed to start using the tool or joining the decision-making process. It also showed that the role of the dialogue mapper should be the role of a person who leads a panel of speakers but should not get involved into the decision-making process because this could result into the dialogue mapper steering the decision making.

During the evaluation, some problems with the user interface of the tool became apparent. These problems are discussed in the limitations and future work section of this chapter. Although most participants indicated that this tool provides benefits and stated that they would use a tool like this in the future, not all of them were positive. Two out of the fourteen participants stated that they did not see any added benefits for their productivity by using this tool. One participant stated that he would use this tool in the future if some changes were introduced.

7.3 Limitations and Future Work

During the evaluation of the tool, it became apparent that some limitations and usability problems exist. These limitations together with what could possibly be done to tackle those limitations in the future are discussed in

this section.

7.3.1 Visualization of Decision Information

During the evaluation of the extended GuideaMaps tool, several participants noted that the organization of decision information on the screen could become overwhelming or cluttered when the decision-map becomes big. The tool gives the users the ability to structure the different elements of a decision-map freely on the screen, but it turned out that this freedom does not result in the most efficient way to structure the information. Some pre-defined structure for the elements could be used so that decision-maps that become big stay orderly and clear. Research could be done to find out the optimal structuring of the elements.

The original issue of the discussion should always be clearly visible at all times to make sure not to lose focus. When creating a new idea and going down the path of this idea by adding more elements to it, the original idea should also be visible, or it should be easy to traverse back to the original idea.

7.3.2 Introducing More Context

More context could be introduced into the decision-making maps by letting users add images, audio, videos or other types of media to the elements in the decision-map. This could provide extra information to each element in the decision-map and add extra context on why decisions were made.

Another way to add more context would be the ability to identify the users participating in the discussion by assigning a color or a name to each participant. The different elements on the decision making map could use this color or name to add information on who took certain decisions.

By introducing a weight to the arguments used, there can be made a distinction between strong and weak arguments. When an idea has five weak pro arguments but one strong con argument the idea may be discarded while in the current implementation it seems like the idea is a good idea because it has many pro arguments.

7.3.3 Extracting more meaningful information

The current implementation of the extended GuideaMaps tool provides data that can be extracted in the form of a time-stamp that is added to a node on the moment of creation. The tool could be further extended to provide more data such as for example the name of the person who brings up a

question, idea or argument during a discussion or the type of stakeholders that participate in the discussion of a particular issue. Adding this extra data does not only introduce more context but also introduces the opportunity to extract more meaningful information from the tool. This data can be extracted and data visualization tools or techniques might be used to study this data. Possible points of interest might be "What types of issues are certain types of stakeholders most involved in?" or "Is there a certain order in which different issues are tackled that seems to be returning over time?"

7.3.4 Web Application

A version of the extended GuideaMaps tool that is used for collaborative decision-making but not in a face-to-face meeting setting could be researched. A web application could be created in which users follow the discussion at home and participate in the decision-making process without being physically in the same room. One person could still function as the dialogue mapper while the other participants see the overview of the decision-map on their own local screen. By creating a web application it would also be possible to use the tool on many different platforms instead of only an iPad tabled device. The question remains whether this would add improvements over the original tool or not.

References

Android and iOS os fragmentation. (n.d.). `http://mattpilz.com/rip-original-ipad-ios-device-fragmentation/`. (Accessed: 2015-06-30)

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, *4*(3), 114–123.

Bolstad, C. A., & Endsley, M. R. (1999). Shared mental models and shared displays: An empirical evaluation of team performance. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 43, pp. 213–217).

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (1998). Extensible markup language (xml). *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210*, *16*.

Brooke, J. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), 4–7.

Brooke, J. (2013). Sus: a retrospective. *Journal of Usability Studies*, *8*(2), 29–40.

Burge, J. E., & Brown, D. C. (2004). An integrated approach for software design checking using design rationale. In *Design computing and cognitionâĂŹ04* (pp. 557–575). Springer.

Burge, J. E., & Brown, D. C. (2006). Rationale-based support for software maintenance. In *Rationale management in software engineering* (pp. 273–296). Springer.

Conklin, E. J., & Yakemovic, K. (1991). A process-oriented approach to design rationale. *Human-Computer Interaction*, *6*(3), 357–391.

Conklin, J. (2003). Dialog mapping: Reflections on an industrial strength case study. In *Visualizing argumentation* (pp. 117–136). Springer.

Conklin, J. (2005). *Dialogue mapping: Building shared understanding of wicked problems*. John Wiley & Sons, Inc.

Conklin, J., & Begeman, M. L. (1988). gibis: A hypertext tool for exploratory policy discussion. *ACM Transactions on Information Systems (TOIS)*, *6*(4), 303–331.

Conklin, J., & Begeman, M. L. (1989). gibis: A tool for all reasons. *Journal of the American Society for Information Science (1986-1998)*, *40*(3), 200.

Conklin, J., Selvin, A., Shum, S. B., & Sierhuis, M. (2001). Facilitated hypertext for collective sensemaking: 15 years on from gibis. In *Pro-*

ceedings of the 12th acm conference on hypertext and hypermedia (pp. 123–124).

Davis, J. H., Brandstätter, H., & Stocker-Kreichgauer, G. (1982). *Group decision making.* European Association of Experimental Social Psychology by Academic Press.

Deip, P., Thesen, A., Motiwalla, J., & Seshardi, N. (1977). Nominal group technique.

De Troyer, O., & Janssens, E. (2014). Supporting the requirement analysis phase for the development of serious games for children. *International Journal of Child-Computer Interaction*, *2*(2), 76–84.

Halpin, T. (2006). Object-role modeling (orm/niam). In *Handbook on architectures of information systems* (pp. 81–103). Springer.

Hess, S., & Jung, J. (2012). Does the ipad add value to business environments? In *Chi'12 extended abstracts on human factors in computing systems* (pp. 335–350).

iOS delegation design pattern. (n.d.). `https://developer.apple.com/library/ios/documentation/General/Conceptual/CocoaEncyclopedia/DelegatesandDataSources/DelegatesandDataSources.html`. (Accessed: 2014-10-30)

Jarczyk, A. P., Löffler, P., & Shipman III, F. M. (1992). Design rationale for software engineering: a survey. In *System sciences, 1992. proceedings of the twenty-fifth hawaii international conference on* (Vol. 2, pp. 577–586).

Kunz, W., & Rittel, H. W. (1970). *Issues as elements of information systems* (Vol. 131). Institute of Urban and Regional Development, University of California Berkeley, California.

Lee, J. (1989). Decision representation language (drl) and its support environment.

Lee, J., & Lai, K.-Y. (1991). What's in design rationale? *Human–Computer Interaction*, *6*(3-4), 251–280.

Linstone, H. A., Turoff, M., et al. (1975). *The delphi method: Techniques and applications* (Vol. 29). Addison-Wesley Reading, MA.

MacLean, A., Young, R. M., Bellotti, V. M., & Moran, T. P. (1991). Questions, options, and criteria: Elements of design space analysis. *Human–computer interaction*, *6*(3-4), 201–250.

McCall, R. J. (1989). Mikroplis: a hypertext system for design. *Design Studies*, *10*(4), 228–238.

McCall, R. J. (1991). Phi: A conceptual foundation for design hypermedia. *Design Studies*, *12*(1), 30–41.

Miller, G. A. (1956). The magical number seven, plus or minus two: some

limits on our capacity for processing information. *Psychological review*, *63*(2), 81.

Muller, M. J., & Kuhn, S. (1993). Participatory design. *Communications of the ACM*, *36*(6), 24–28.

MVC design pattern. (n.d.). `https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller`. (Accessed: 2014-10-30)

Okada, A., & BuCkiNGhAm Shum, S. (2006). Knowledge mapping with compendium in academic research and online education.

Osborn, A. F. (1953). Applied imagination.

Ozok, A. A., Benson, D., Chakraborty, J., & Norcio, A. F. (2008). A comparative study between tablet and laptop pcs: User satisfaction and preferences. *Intl. Journal of human–computer interaction*, *24*(3), 329–352.

Pavlovych, A., & Stuerzlinger, W. (2008). Effect of screen configuration and interaction devices in shared display groupware. In *Proceedings of the 3rd acm international workshop on human-centered computing* (pp. 49–56).

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2006). The design science research process: a model for producing and presenting information systems research. In *Proceedings of the first international conference on design science research in information systems and technology (desrist 2006)* (pp. 83–106).

Shum, S. B., & Hammond, N. (1994). Argumentation-based design rationale: what use at what cost? *International Journal of Human-Computer Studies*, *40*(4), 603–652.

Shum, S. J. B., Selvin, A. M., Sierhuis, M., Conklin, J., Haley, C. B., & Nuseibeh, B. (2006). Hypermedia support for argumentation-based rationale. In *Rationale management in software engineering* (pp. 111–132). Springer.

Takeda, H., Veerkamp, P., & Yoshikawa, H. (1990). Modeling design process. *AI magazine*, *11*(4), 37.

Toulmin, S. E. (2003). *The uses of argument.* Cambridge University Press.

VanLehn, K. A. (1985). *Theory reform caused by an argumentation tool.* (Tech. Rep.). DTIC Document.

A

Evaluation forms

SUS Questionnaire

1. **I think that I would like to use this system frequently**
   Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

2. **I found the system unnecessarily complex**
   Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

3. **I thought the system was easy to use**
   Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

4. **I think that I would need the support of a technical person to be able to use this system**
   Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

5. **I found the various functions in this system were well integrated**
   Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

6. **I thought there was too much inconsistency in this system**
   Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

Figure A.1: The SUS questionnaire (page 1)

7. **I would imagine that most people would learn to use this system very quickly**
Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

8. **I found the system very cumbersome to use**
Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

9. **I felt very confident using the system**
Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

10. **I needed to learn a lot of things before I could get going with this system**
Mark only one oval.

	1	2	3	4	5	
strongly disagree	◯	◯	◯	◯	◯	strongly agree

Figure A.2: The SUS questionnaire (page 2)

Opinion form

1. **Was the organization of decisions on the screen clear?**
   Check all that apply.
   - ☐ Yes
   - ☐ No
   - ☐ Other: .....

2. **Was the meaning of the different colors clear?**
   Check all that apply.
   - ☐ Yes
   - ☐ No
   - ☐ Other: .....

3. **Do you think the discussion was easy to follow on the screen?**
   Check all that apply.
   - ☐ Yes
   - ☐ No
   - ☐ Other: .....

4. **Is there something you did not like about the platform, and would like to change?**

   .....

   .....

   .....

   .....

   .....

5. **Do you think this software provides added benefits to regular requirement analysis/meetings?**

   .....

   .....

   .....

   .....

6. **Would you want to use this software in the future?**
   Check all that apply.
   - ☐ Yes
   - ☐ No
   - ☐ Other: .....

Figure A.3: The participants' opinion form

B

Evaluation presentation

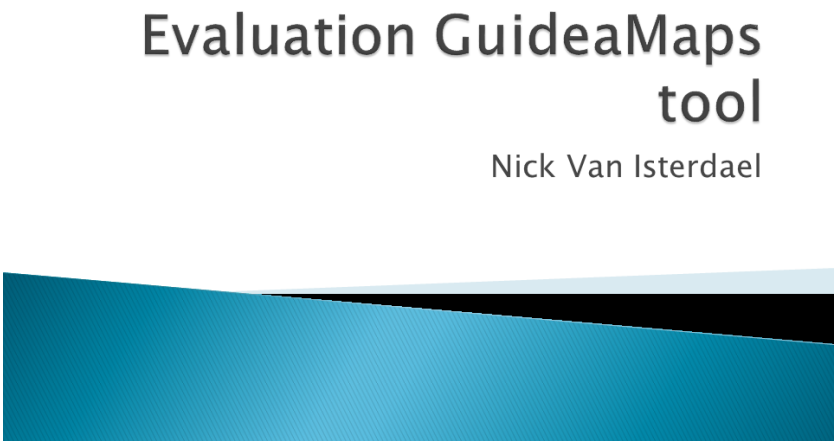

Figure B.1: The evaluation presentation (slide 1)

Overview

- Background
- Experiment
- Tool introduction

Figure B.2: The evaluation presentation (slide 2)

Background

- Original tool used for requirement analysis
  - "Requirementsanalyse in software engineering en systems engineering is het bepalen en overwegen van de requirements van een nieuw of te wijzigen product, rekening houdend met mogelijke conflicterende vereisten van de betrokken stakeholders".

Figure B.3: The evaluation presentation (slide 3)

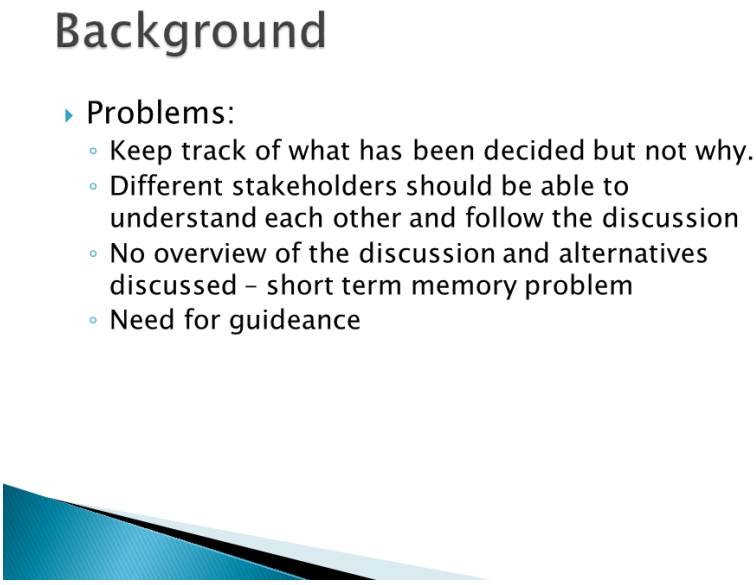

Figure B.4: The evaluation presentation (slide 4)

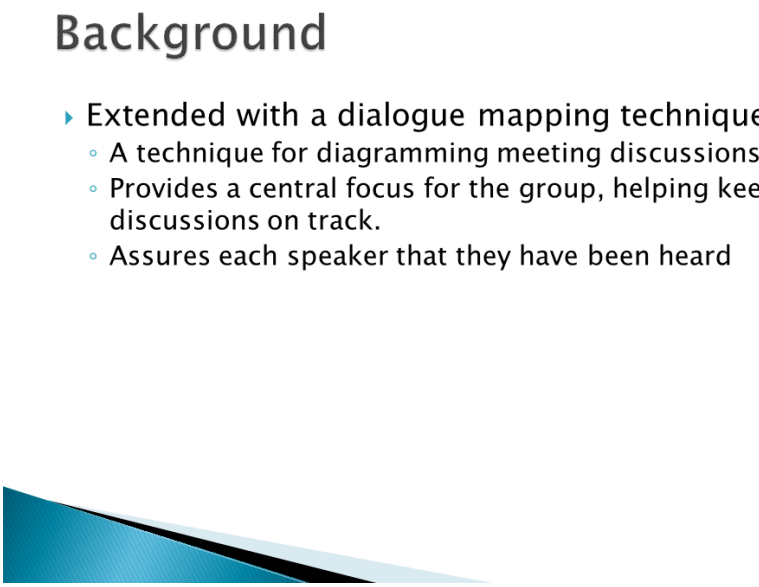

Figure B.5: The evaluation presentation (slide 5)

Background

- Requirements for dialogue mapping
  - Shared display
  - Dialogue mapper
  - Graphical language

Figure B.6: The evaluation presentation (slide 6)

Background

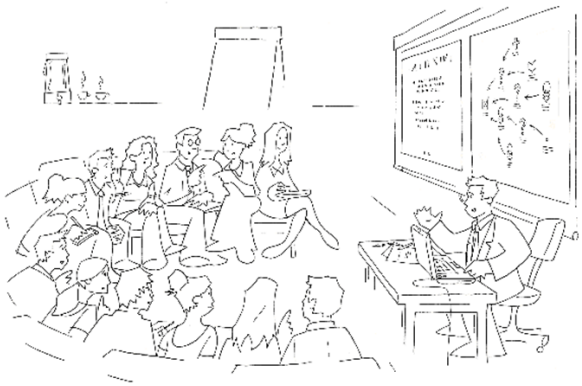

Figure B.7: The evaluation presentation (slide 7)

Brief introduction

- IBIS framework
  - Notation consists of three elements
    - Issues (or questions)
    - Positions (or ideas)
    - Arguments (pro or con)
  - Grammar can be summerized in three simple rules

Figure B.8: The evaluation presentation (slide 8)

Brief introduction

- Example

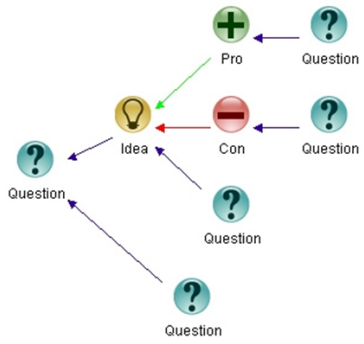

Figure B.9: The evaluation presentation (slide 9)

Brief introduction

- Why IBIS?
  - Simple en intuitive, requires no training
  - Expressive enough to tackle any problem

Figure B.10: The evaluation presentation (slide 10)

Experiment

- Groups of atleast 3 people
- One dialogue mapper
- Use case – create game
- Questionnaire – SUS usability evaluation
- Evaluate opnions

Figure B.11: The evaluation presentation (slide 11)

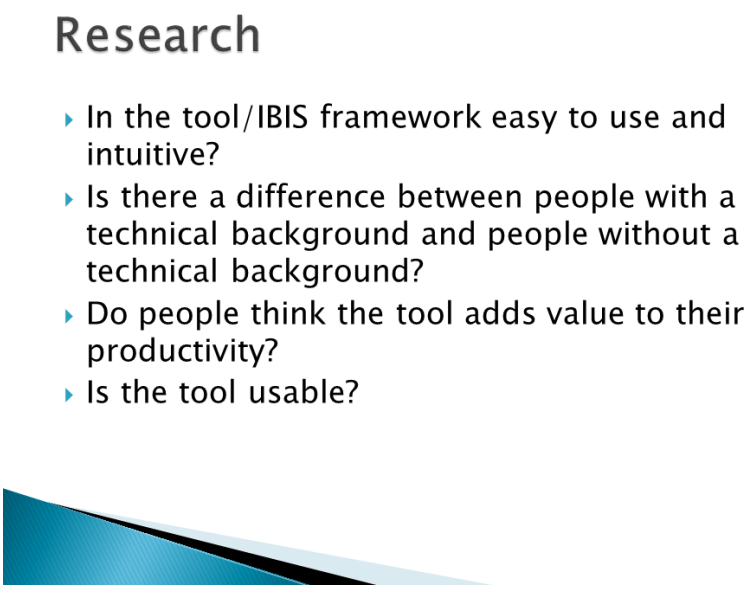

Figure B.12: The evaluation presentation (slide 12)