Vrije Universiteit Brussel

FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRONICS AND INFORMATICS

A Collaborative iPad Tool for Requirement Elicitation

Master thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Applied Sciences and Engineering: Applied Computer Science

Johannes Licha

Promoter: Prof. Dr. Olga De Troyer
Advisor: Erik Janssens

Academic year 2014-2015

Abstract

Requirement elicitation is an essential first step in the development of any software project. It is a genuine collaborative task, which usually involves technical and non-technical experts with different backgrounds. It can be challenging to find a common language and notation to discuss ideas and capture problems and solutions. Inefficient communication between stakeholders is a typical problem.

This thesis investigates the potential of using tablets in meetings for a domain-specific requirement elicitation application, i.e. GuideaMaps. The focus lies on supporting collaboration by connecting tablets and allowing users to work in a common workspace.

The characteristics of such a collaborative application, which supports users in the process of domain-specific requirement elicitation were defined as a set of functional and usability requirements. An important component of this enterprise was investigating related scientific work and analyzing available applications. This revealed that existing applications do not fulfill those requirements.

These findings led to the development of a new prototype application based on an existing single user version of GuideaMaps. The GuideaMaps tool takes a guided ideation approach on requirement elicitation by visualizing the users decisions in a hierarchical structure similar to a MindMap. The main objective was to modify and extend the existing tool in order to support collaborative group work.

The new application is designed to be used in individual work as well as in a group meeting context. It allows users to develop their ideas in private, but also to share their work with others, if and when they wish. It supports a group in finding a common solution by allowing comparing different contributions and pointing out potential conflicts between ideas from different users.

An informal case study was carried out to obtain initial feedback on the usability of the new application. It revealed some problems and limitations which could be taken as input for an improved version and for possible future work.

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

I would like to express my gratitude to my promoter Prof. Dr. Olga De Troyer for her useful comments, remarks and engagement through the learning process of this master thesis.

Thank you Mama and Ewald for helping a late bloomer to discover his potential and the constant support over the last years.

Thanks to my father, who didn't know anything about computers, but aroused my interest in science and encouraged me try to understand how stuff works.

Thank you Laura, for helping me to concentrate when I needed to and for distraction when I was fed up with computers.

Contents

List of Figures

1 Introduction

1.1 Context and Problem Statement

As one of the first steps, requirement elicitation is an essential part of the development cycle of any software project. In order to sufficiently consider all relevant aspects it is preferable to involve other parties beside programmers at an early stage of the development process. Usually, during requirement elicitation, different stakeholders are consulted. Therefore requirement elicitation is a highly collaborative task potentially including not only developers and designers, but also domain experts, customers, users, and other relevant stakeholders. One main issue that naturally emerges from the diversity of the participants' backgrounds is the need to find a common language to discuss the various aspects of a project and a means of capturing ideas, problems and solutions. Another typical problem during this planing phase is that the notations used are often either too imprecise or too technical (Al-Rawas & Easterbrook, 1996).

As part of his master thesis (Janssens, 2013), Eric Janssens created an application for the iPad, called GuideaMaps, which takes an intuitive approach, i.e. guided ideation, towards domain-specific software requirement elicitation. This is achieved by visualizing issues to be discussed and decisions taken in a meeting, using a predefined structure similar to a MindMap (Buzan,

2004). This tool successfully provides a space to document, in an easy and intuitive way, ideas, problems, and decisions taken. It can be used by IT professionals as well as less technically oriented people, whose expertise lies in other fields. However, this tool is a single user tool and lacks support for collaboration between people. Since ineffective communication between stakeholders is a common problem in this field (Al-Rawas & Easterbrook, 1996), collaborative aspects are an interesting topic to investigate for the further development of GuideaMaps.

New forms of collaborative work, using IT-systems, have been the subject of research for decades, but the development of new devices, user interfaces and network technologies in recent years finally made it possible to develop software that naturally integrates in collaborative everyday working practices like meetings. This has the potential of drastically increasing efficiency, delivered at a reasonable price. For example, suitable mobile devices (like tablets), equipped with all necessary hardware features, are now readily available and affordable. Wireless network technology, touch screens with high resolution and sensitivity and long-lasting, light-weight battery technology make them suitable for meetings and they are less obtrusive than laptops.

Incorporating new techniques for collaborative group work into the guided ideation approach for requirement elicitation is a promising combination, which could support the collaborative nature of this task. Extending the GuideaMaps application with collaborative features should facilitate discussion and document decisions taken by a group of people during a meeting, and should provide a clear and well-structured means of communication between all participants. Even though requirement elicitation is a genuinely collaborative task, ideas are often developed in private as preparation for a meeting. After presenting their ideas to the group, users can compare their contributions and weigh up all the advantages and disadvantages. After a phase of discussing, merging and deleting ideas, the group may come to a common solution. During that process the tool should support the group, e.g. by pointing out potential conflicts between mutually exclusive ideas from different users.

1.2 Research Goals

Based on the formulated problem statement, the main research goal is to extend the GuideaMaps application with the addition of collaborative features.

This is a so-called practical research problem (Wieringa, 2009). In order to solve this practical problem, we first need to addresses a more general research question:

Q0: How can tablet software support technical and non-technical stakeholders when working collaboratively in the field of domain-specific software requirement elicitation?

In order to investigate this question in more detail and illustrate the main points of interest, it can be further refined in the following sub-questions:

Q1: Which approaches and techniques already exist to address Q0?

Tablets like iPads are handy tools in meetings and different applications can be found on Apple's iOS App Store. But also software on other platforms and various browser applications are available and will be taken into consideration.

Q2: How can contributions from different users be visualized and compared during a meeting?

Q3: How can contributions from different users be merged during a meeting?

Q4: How can conflicts between mutually exclusive ideas from different users be resolved during a meeting?

Q5: How can the users keep track of the progress and current state of work?

Requirement elicitation is a complex task. Requirements may be required for various aspects of the project. In the process of discussing ideas and proposals from different stakeholders, the users can quickly lose sight of the overview of what has been agreed on and which aspects are still pending.

1.3 Proposed Solution

The existing GuideaMaps tool is used as starting point to develop a collaborative version.

The new collaborative tool will allow stakeholders to connect their devices during a meeting using wireless technology. Each user has a private workspace to develop ideas in advance or during the meeting. All participants share a common workspace where users present their own ideas and can compare

them to contributions from other users. At the beginning of a meeting users can synchronize devices to upload their own contributions and download the newest version of the shared workspace. At the end of a meeting the latest state of the shared workspace is stored on all devices.

The tool supports the group in finding a solution efficiently by pointing out potential conflicts between ideas from different stakeholders.

The tool allows the user to show the shared workspace on a screen. This visualization of all common ideas can be displayed on a big screen or projector visible for everyone. It helps users to keep track of the current state of work.

1.4 Research Methodology

The design science research methodology (DSRM) "incorporates principles, practices and procedures required to carry out [..] research" (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). This methodology will serve as a framework to obtain our research goal in a structured manner. DSRM prescribes a number of steps which are discussed below.

1.4.1 Problem Identification and Motivation

In sections 1.1 and 1.2 the specific research problem and its motivations have been identified, and the value of a solution has been justified.

1.4.2 Definition of the Objectives for a Solution

The objectives for a solution are briefly described in section 1.3, but will be captured in more detail by defining a number of functional requirements and usability requirements in chapter 2. In order to answer Q1, a study of the scientific literature in the field of requirement elicitation with a focus on collaborative tools has been carried out and is described in chapter 3. Furthermore the functionality of a number of existing applications is compared to the defined requirements.

As the findings were that no available application sufficiently fulfills those requirements, we decided to develop a prototype of a new tool.

1.4.3 Design and Development of a Solution

As no available application sufficiently fulfills our requirements, we decided to design and develop our own solution. The collaborative tool is designed and implemented as an iOS application based on the existing GuideaMaps application.

The main issues which have to be addressed are:

- A basic precondition for all collaborative features is the existence of an application specific network to connect the devices and enable communication between the users.
- Development of a user interface for visualizing contributions from different users in a way that makes it easy to compare and merge them (Q2 and Q3).
- The application has to analyze the users contributions and point out conflicts between choices of different users (Q4).
- The application must be able to create a visualization of the shared workspace in order to support the user in keeping track of the current state of work (Q5).
- The application needs to be implemented using hardware integrated in iPads and frameworks available on iOS.

1.4.4 Evaluation

To obtain initial feedback on the usability of the tool, a pilot evaluation was carried out. It was conducted as an informal case study. Five users with technical/pedagogical background were asked to develop an initial set of requirements for a serious game. In the first phase the participants prepared their ideas individually. In the second phase, their contributions were compared and merged to find a common solution. Data was collected through observation during the meeting and direct feedback from the users. The goal of this evaluation was to obtain initial feedback on the usability of the tool, with a focus on the collaborative aspects.

1.5 Structure of the Thesis

In chapter 2 functional requirements and usability requirements are defined. The new application has to fulfill those requirements.

Chapter 3 further investigates Q1. A review of available applications reveals a set of interesting features, but also their problems and limitations. The chapter provides an overview of related research work, which could be useful to answer the formulated research questions. None of the analyzed applications fulfilled the formulated requirements, so a prototype of a new applications has been developed.

The application GuideaMaps, presented in chapter 4, is used as a basis for the new prototype.

Chapter 5 introduces the developed prototype application. It describes user interface and design principles, referring to usability and functional requirements and addresses questions Q2 - Q5.

Chapter 6 contains technical details about the implementation of the developed tool.

Chapter 7 documents the evaluation of the new application and proposes possible future work.

The last chapter summarizes the finding of this research thesis and documents the conclusion.

2 Requirements

The following functional requirements and usability requirements for a collaborative requirement elicitation tool are defined below.

2.1 Functional Requirements

F1: The tool should support a single user working mode without network connection

F2: The tool should allow users to connect their tablets and work in a shared space

F3: Users should be able to develop ideas in private and share them with the group at any time

F4: Users should see their own ideas and ideas other users have shared in parallel, in order to be able to compare them

F5: The tool should point out potential conflicts between mutually exclusive ideas from one user or from different users

F6: The tool should be able to save the current state of the shared space

F7: The tool should allow several iterations of single and group working sessions using the same shared space

F8: The tool should be able to present a visualization of the shared space (on a screen/projector)

F9: The tool should be able to connect devices without the need of external network connections

F10: Users should be able to join or leave a group meeting at any time

2.2 Usability Requirements

U1 The tool should be usable by IT-professionals as well as non-technical domain experts without the need to read a manual

U2 Changes in the shared space should be instantly visible for all users

U3 The source of contributions should be visible for the group

3

Related Work

This chapter provides an overview of available tools and research literature in the context of collaborative requirement elicitation. The analysis of existing work will help to answer the research question Q1: *Which approaches and techniques already exist to address Q0?*. We first discuss existing collaborative ideation tools (section 3.1). Next, we discuss related research work (section 3.2).

3.1 Collaborative Ideation Tools

There are a number of applications for collaborative mind mapping available on Apple's App-Store and through different sources on the Internet. Even though those tools are not customized for software requirement elicitation and some are not specifically developed for tablets, the problems they address in the general context of collaborative mind mapping and their solutions are of interest and aspects may be adaptable for our purpose.

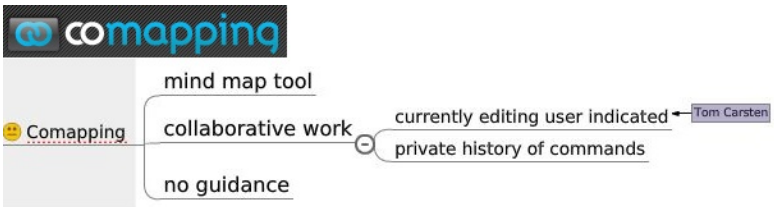

Figure 3.1: Comapping

3.1.1 Comapping

Comapping[1] is a collaborative mind mapping tool (see figure 3.1), which allows several users to edit a shared idea space as a hosted web service. The ownership of an idea is not constantly visible, but a small label shows who is currently editing which part of the map. All changes are instantly updated and visible to all users, which prevents the occurrence of any conflicts. Users can undo and redo commands using their private editing history.

3.1.2 iBrainstorm

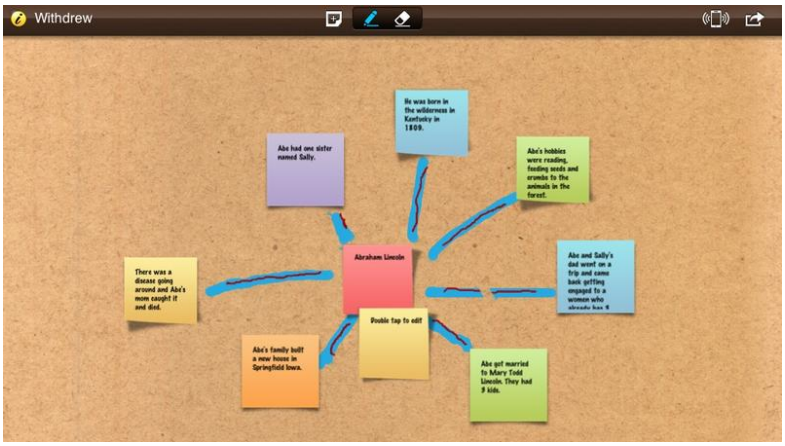

Figure 3.2: iBrainstorm

iBrainstorm[2] (see figure 3.2) is a collaborative multi-device brainstorming tool. Ideas are represented as postit-like notes, which can be edited using different colours and sizes and moved around. The background can be used as a sketch board and allows the user to loosely structure ideas and indicate

[1] `www.comapping.com`
[2] `www.ibrainstormapp.com`

dependencies. Ideas can be shared with the group by moving them to a group-icon in the corner. The collaborative aspects of this tool are very limited. Shared ideas can not be made private again and deletion does not affect ideas on other devices. Sketches, which potentially hold information about structure and dependencies between ideas cannot be shared. Also, the tool does not provide any guidance functionality.

3.1.3 Lucidchart

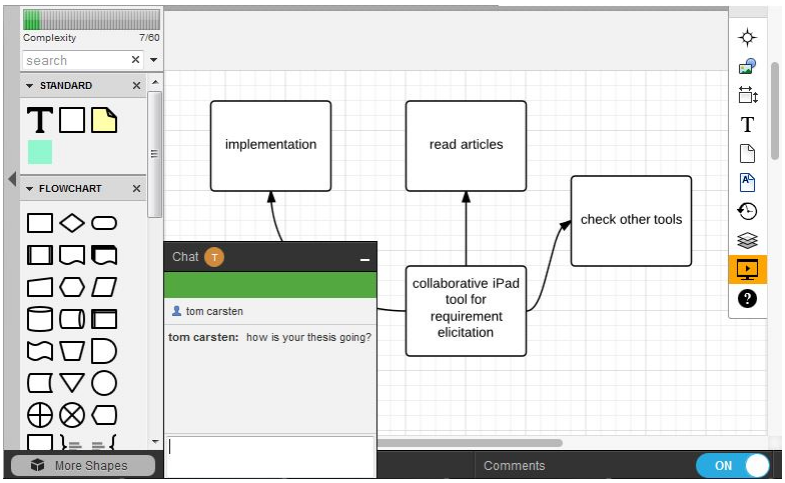

Figure 3.3: Diagram in Lucidchart with chat window

Lucidchart[3] (see figure 3.3) is a realtime collaboration tool for various forms of diagrams (including mind maps). It is not only suitable for meetings, but also for online collaboration when users are working in different locations because of the chat functionality. An interesting feature is the revision history. The tool lists previous versions of the document showing the time and the user who made the changes. When clicking on an older version, the changes are highlighted in the diagram. The user can go back to any point in history and either restore that state of the document or create a new document from that state (fork).

3.1.4 Mindmeister

Mindmeister[4] (see figure 3.4) is a browser application for collaborative mind mapping. When different users work simultaneously on one map, a small

[3] `www.lucidchart.com`
[4] `www.mindmeister.com`

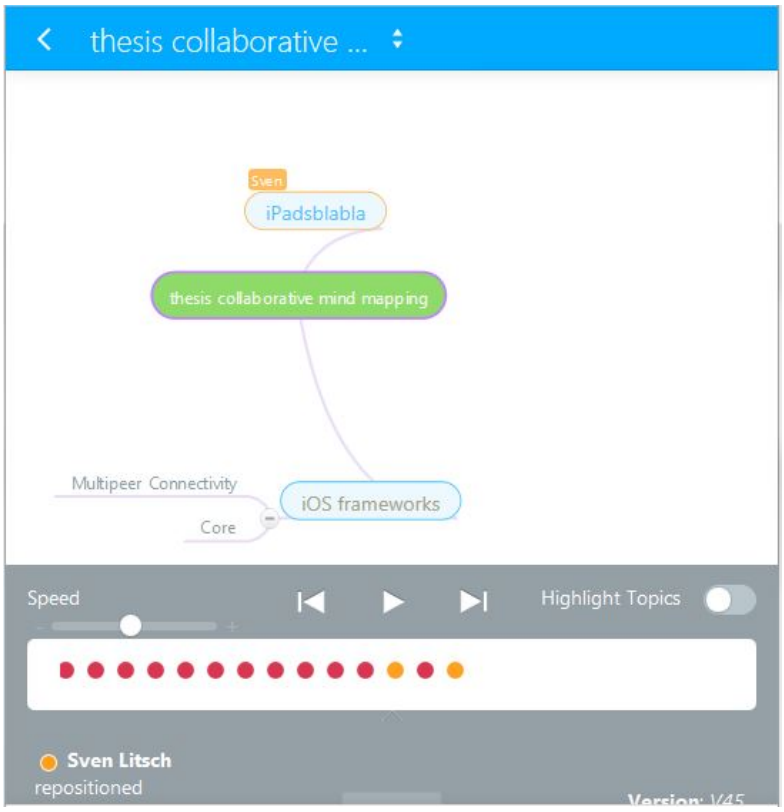

Figure 3.4: Mindmap in mindmeister with history on the bottom

label in the corner of an idea shows which user is currently editing an idea. However, several users can edit one idea at the same time, which can be problematic. The application takes the most simplistic approach to handle conflicts. The last user who ends the editing process will overwrite all previous changes. The history of the mind map can be shown at the bottom of the screen. The history is represented by a line of dots, where every dot stands for one change. The user who changed the document is indicated by a color. The time and date is shown when hovering over a dot. The user can click on a dot and see the state of the mind map at that time. When clicking on play, the evolution of the mind map is shown in fast motion.

3.1.5 mind42

mind42[5] allows sharing maps with other users and supports real-time collaboration. It provides a very simple history with undo-redo functionality. The

[5] www.mind42.com

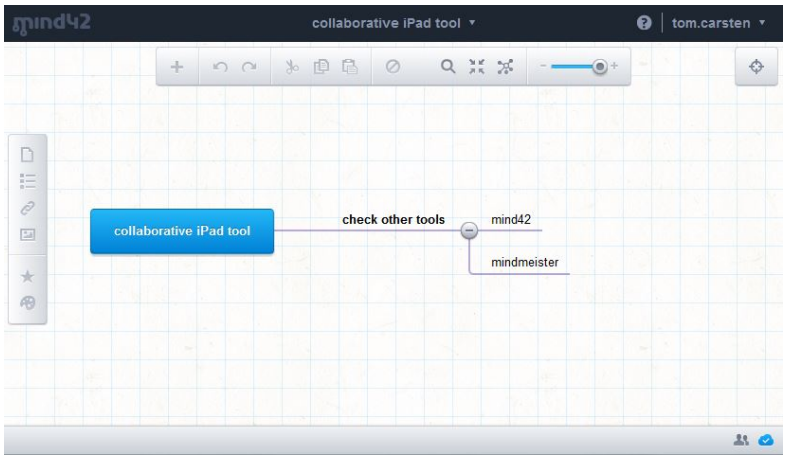

Figure 3.5: Mindmap in mind42

tool does not indicate if other users are currently editing ideas which leads to problems if several users are working on the same comment.

3.2 Related Scientific Work

This section is dedicated to related work from research literature in the field of collaborative work for requirement elicitation. This context is limited, particularly in combination with a guided ideation approach.

We first briefly explain two concepts commonly used in such research works.

- Production blocking: Only one member of the group can present his idea at a time - other participants are blocked. These participants might forget or suppress their ideas if they can't contribute them immediately.

- Evaluation apprehension: members of the group withhold ideas because they fear to be evaluated or judged by others.

(Shih, Nguyen, Hirano, Redmiles, & Hayes, 2009) investigated brainstorming in a group setting using a real-time collaborative mind-mapping application, compared to the classical approach using a white board. Users shared a single virtual mind map displayed on a big screen. Unlike when using a white board, tablet users could contribute ideas at any time, without having to go through a facilitator. This affected the overall group dynamic positively and potentially created more egalitarian participation. But the fact that all input

was visible to all members could also lead to evaluation apprehension. An interesting feature they introduced was color-coding to distinguish different user's contributions.

(Prante, Magerkurth, & Streitz, 2002) compared a collaborative mind mapping system and two collaborative whiteboard editors. The mind mapping application uses a turn-taking working mode where all participants can instantly see all changes. However, only one participant can alter the idea space at a time, resembling a traditional face-to-face meeting situation where only one person can speak at a time. This working mode reduced output due to production blocking. From this study, functional and usability requirements were drawn up for developing "software for groups to support idea finding", e.g. "prevention of turn-taking", "structuring the idea space" - as this "seems to foster the group's creative performance". Based on those findings, a tool suite was developed, including an application for mobile devices, which allows asynchronous pre- and postprocessing of meetings. Users can prepare ideas in advance and add them to the public idea space at any point in time during a meeting.

(Geisser & Hildenbrand, 2006) introduced a systematic and well structured method for iterative, collaborative requirement elicitation followed by a decision supported analysis of cost and values in order to prioritize and select requirements for the final implementation. The focus lies on asynchronous contribution in a geographically distributed working mode. Requirements are arranged in sets and their interdependencies, e.g. requirement A is prerequisite for B, are captured in a graph-like structure. The process of selecting/discarding ideas (in order to agree on a final set) is implemented using pairwise comparisons to establish priorities among requirements (analytic hierarchy process). However, this complex approach with several predefined phases requires prior training for participants.

(Zarinah & Siti Salwah, 2009) also proposed a structured method for collaborative requirement elicitation that is split in different phases, potentially subdivided in several steps, which are performed iteratively. When a project is set up, a dedicated facilitator is appointed who takes a leading role in the following three main phases: identification, elaboration and integration. This rigid structure is supposed to prevent incomplete, insufficient requirement collection.

(Jaafar, Atan, & Nazatul, 2011) introduced a web based prototype for a

collaborative mind map tool for software requirement elicitation. The application supports synchronous and asynchronous contribution and allows users to manage the accessibility of information by keeping ideas private or sharing them with other participants. Users start with an empty map and therefore rely on the knowledge and experience of experts to make sure all relevant aspects of the project are covered and the process ends with a complete set of requirements. The tool does not provide any guidance or predefined domain specific structure to facilitate the requirement elicitation process.

3.3 Summary

A wide range of collaborative Mind Mapping tools are available. But no applications for requirement elicitation using a guided ideation approach have been found.

A few general ideas and some nifty features from the set of analyzed applications can be used as inspiration for the design of a new tool. Some interesting ideas and concepts generated from this analysis are summarized in the following list:

- In order to work smoothly it is essential that changes by one user are immediately visible for the group
- The ownership of contributions should be visible for the group
- Ideas can be shifted between a private and a shared workspace
- Every user can contribute ideas at any time without going through a facilitator, to prevent production blocking and evaluation apprehension
- The tool should support individual and collaborative work and allow a fluent transition between the two working modes. This way of working enables users to prepare ideas and participate in a meeting with the same application.
- A simple design which does not demand any training is desirable
- A history allows more agility in the discussion of ideas. It is not necessarily connected to group work, but a collaborative requirement elicitation tool would certainly benefit from it. Users wouldn't hesitate to make changes to the public workspace, because they can always be undone and redone. However, implementing a history for GuideaMaps

goes beyond the scope of this thesis and will rather be the subject of future work.

None of the tools found in this study fulfill the requirements as defined in section 2. Therefore a prototype of a new application has been developed.

4
Background - GuideaMaps

GuideaMaps is an iPad application for domain-specific requirement elicitation, developed by Eric Janssens, as part of his master thesis and was originally applied to the field of serious games. The application can be used on a single device, but does not offer networking functionality or support for collaborative working.

In this chapter, we describe the principles used by GuideaMaps, as well as its user interface and main features.

4.1 A Decision Model using feature-based Modeling

GuideaMaps was originally developed for the requirement elicitation process for serious games (De Troyer & Janssens, 2014b) but can also be used for the requirement elicitation for other domains (De Troyer & Janssens, 2014a). For specific domains, such as serious games or web systems, it is largely known which types of requirements have to be formulated. During the requirement elicitation phase for such domain-specific software, all stakeholders have to go through the various aspects of the software, decide on different options to choose from and consider their dependencies. In general, the problem is

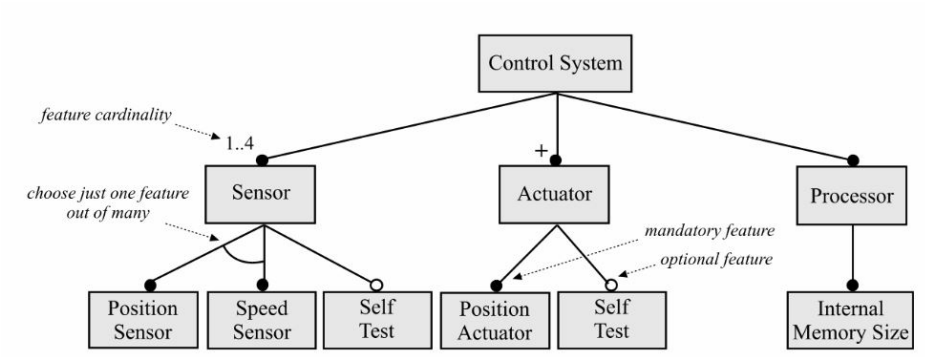

Figure 4.1: Example of a feature model diagram

that the types of requirements to be considered is known, but not readily available. In order to offer some guidance to the user, the different types of requirements to be considered, as well as options, alternatives and dependencies should be provided to the user in a structured way. For this, GuideaMaps is using a feature modeling technique. This technique is based on the definition of a kind of meta model, also called decision model. This model captures general characteristics of the software in a particular domain including their dependencies.

In (Kang, Cohen, Hess, Novak, & Peterson, 1990) the concept of Feature Oriented Domain Analysis (FODA) for software development has been introduced, including feature modeling. Today feature modeling is one of the most commonly used techniques for modeling variability in software, i.e. to express the common and variable features. Feature model diagrams like Figure 4.1 (Cechticky, Pasetti, Rohlik, & Schaufelberger, 2004) are a visualization of the features in software and their possible options as an and-or tree. Every node demands a decision and every branch of the tree represents one of the possible options. It is possible to express constraints like mandatory or optional features, cardinality etc. The concept of feature modeling was adapted to represent the possible requirements and associated choices a user faces, when defining software for a specific domain such as a serious game.

In GuideaMaps, the feature models are defined using XML. The Extensible Markup Language (XML) is the definition of a textual data format to represent information in a structured way, which is human-readable as well as machine-readable. It has been defined by W3C in the XML 1.0 specification and other related specifications as an open standard.

A meta model in GuideaMaps has a model name, author, comment and contains various features (called Meta-Guideas). A feature represents one aspect of the software and can be further refined using sub-features. The feature description gives additional information on which aspect of the software is covered by this feature.

A feature tagged as *abstract* can be seen as a placeholder for different options the user has to chose from. These options are the children of the *abstract* feature. They *extend* their *abstract* parent. The *cardinality* of an *abstract* feature defines upper and lower bounds for the number of extending children the user can choose. Each feature contains a unique *id* within the model. If a feature only makes sense in combination with or relies on another feature, the *requires* tag can be used to refer to its *id*. If a feature cannot be combined with another feature in the model, that can be expressed using the *excludes* tag. Those two keywords introduce the notion of dependencies between different features across the meta model and specify how features affect other features.

The following code shows a shortened version of a meta model for serious games.

```
<?xml version="1.0" encoding="UTF-8"?>
<product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.codefromtheattic.com/
        featureModel.xsd">
    <model>Serious Game Requirements</model>
    <author>Olga De Troyer</author>
    <comment>Test definition process for a Serious Game.</comment>
    <feature id="root">
        <name>My Serious Game</name>
        <feature id="context">
            <name>Context of Use</name>
            <feature id="devices" abstract="TRUE">
                <name>Target Devices</name>
                <description>Select one or more target devices</description>
                <cardinality min="1" max="5"/>
                <feature id="device1" extends="TRUE">
                    <name>PC</name>
                </feature>
                <feature id="device2" extends="TRUE">
                    <name>Laptop</name>
                    <requires>goal2</requires>
                </feature>
                <feature id="device3" extends="TRUE">
                    <name>Mobile</name>
                </feature>
                <feature id="device4" extends="TRUE">
                    <name>Web</name>
                </feature>
                <feature id="device5" extends="TRUE">
                    <name>Kinect</name>
```

```
                <excludes>context2</excludes>
            </feature>
        </feature>
    </feature>
    ...
```

The *abstract* feature "Target Devices" allows the user to choose from a range of devices like "PC", "Laptop", etc. The game must use at least one of these devices (*cardinality min="1"*), but not more that five (*cardinality max="5"*). If the game runs on a "Laptop", one of the "pedagogical goals" must be "cognitive problem solving" (*id=goal2*, not in the example). If the game is using a "Kinect", it cannot be used in the "learing context school" (*id=context2*, not in the example).

As mentioned, the meta model defines all possible options the user can choose from, including their constraints and dependencies. During the requirement elicitation process, the user has to take decisions on all mandatory aspects provided and in this way a specific configuration of the meta model is assembled. That configuration, containing all decisions taken by the user, is represented as another XML-based model called "application model". The following code shows a shortened version of an example application model for the above meta model.

```
<?xml version="1.0" encoding="UTF-8"?>
<guideaModel metaModel="com.codefromtheattic.seriousgames">
    <guidea id="root" xPos="77.000000" yPos="396.000000" color="0" expanded=
        "TRUE">
        <comment>General comment about the game</comment>
        <guidea id="context" xPos="260.000000" yPos="615.000000" color="0">
            <guidea id="device4" xPos="82.000000" yPos="640.000000" color="0
                ">
                <comment>makes the game available for a broad audience</
                    comment>
            </guidea>
            <guidea id="device1" xPos="20.000000" yPos="546.000000" color="0
                ">
            </guidea>
            <guidea id="device5" xPos="60.000000" yPos="600.000000" color="0
                ">
                <comment>this devices could be optional</comment>
            </guidea>
            <guidea id="device2" xPos="40.000000" yPos="570.000000" color="0
                ">
            </guidea>
        </guidea>
    </guidea>
</guideaModel>
```

Every application model has a unique reference to its meta model, therefore the name of the meta model should be unique. Each Guidea in the appli-

cation model refers to a feature (Meta-Guidea) from the meta model via its unique *id*. A Guidea in the application model also contains visual information like the x/y position on the screen, the color of the Guidea, and if it is expanded respectively if its children are shown. The user is able to add comments to Guideas to document decisions taken.

The XML application model is defined by the following XSD:

```
<!--W3C Schema generated by XML Spy v4.4 U (http://www.xmlspy.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="
    qualified">
    <xs:complexType name="guideaType">
        <xs:sequence>
            <xs:element name="comment" type="xs:string" minOccurs="0"/>
            <xs:element name="guidea" type="guideaType" minOccurs="0"
                maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="xPos" type="xs:float" use="required"/>
        <xs:attribute name="yPos" type="xs:float" use="required"/>
        <xs:attribute name="color" type="xs:boolean" use="required"/>
        <xs:attribute name="expanded" type="xs:string"/>
    </xs:complexType>
    <xs:element name="guideaModel">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="guidea" type="guideaType"/>
            </xs:sequence>
            <xs:attribute name="metaModel" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

4.2 GuideaMaps - UI and Features

The feature-based meta models provide the foundation to capture reoccurring aspects and patterns of requirements, which can be used as a thread by the user. However, true feature models are too complex and not appropriate for casual users. In order to facilitate the requirement elicitation process for less technically versed users, GuideaMaps is visualizing the information in a more intuitive hierarchical structure similar to a mind map.

Guideas are visualized as rectangular boxes with rounded corners (see Figure 4.2). The name of a Guidea is on top of the box and the description is beneath.

The user can select a Guidea by tapping it with a finger. He can move a Guidea and rearrange the layout via dragging. The whole graph can be

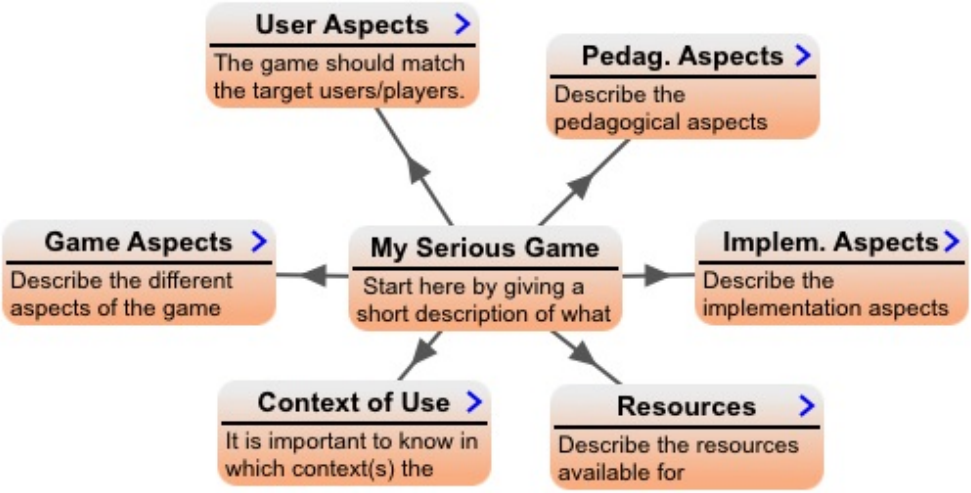

Figure 4.2: Visualization of a feature model in GuideaMaps

moved via dragging on the background outside any Guidea.

Parent-child relationships are indicated by a line with an arrow pointing from the parent to the child. For small graphs like in Figure 4.2 the overall structure is clear, but for large graphs it can become quite messy. In order to not overwhelm the user, parts of the graph are usually hidden and can be expanded if necessary. If a Guidea has hidden children, this is indicated by a small blue arrow in the upper right corner.

4.2.1 Optional Guideas

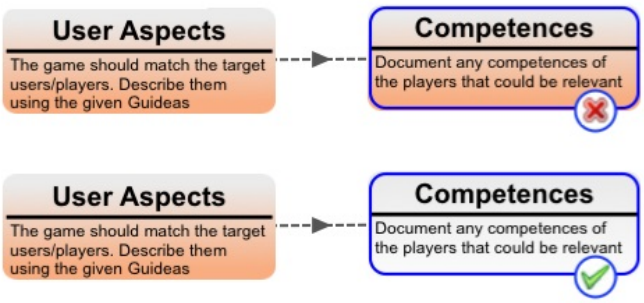

Figure 4.3: Optional Guidea enabled (top) and disabled (bottom)

Guideas can be optional, indicated by a dashed line (see figure 4.3) connecting the parent and an optional child Guidea. When an optional Guidea is selected, a button in the lower right corner allows the user to enable/disable it. Disabled Guideas are colored gray (see lower part of figure 4.3).

4.2.2 Abstract Guideas

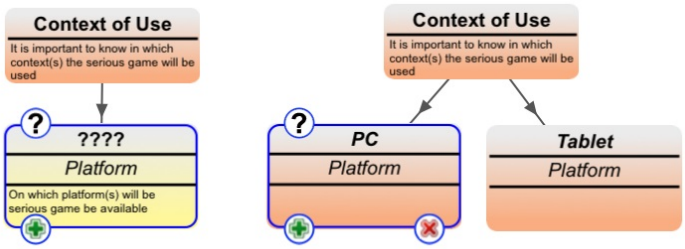

Figure 4.4: Abstract Guidea before any option is selected (left) and after two options have been selected (right).

As mentioned, abstract Guideas act as a placeholder for a range of options, the user can choose from. Contrary to normal Guideas, abstract Guideas consist of three compartments (see figure 4.4). On the top, the selected option is shown. If no option is selected yet, the top compartment contains **????** and the Guidea is colored yellow to draw the users attention to this outstanding decision. In the middle, the name of the abstract Guidea is shown. Similar to normal Guideas, a description is shown on the bottom. When the Guidea is selected, a button on the right upper corner allows the user to choose a concrete option (one of the children of the abstract Guidea). If the maximal number of options is not exceeded (referring to the decision model: *<cardinality min="1" max="5">*), a button in the lower left corner can be used to add another box representing the same abstract Guidea. That allows the user to select another option for the same abstract Guidea.
When clicking the *?* button on an abstract Guidea, a popover menu with all possible options appears (see figure 4.5). If these options require or exclude other Guideas (referring to the decision model: *<requires>* and *<excludes>*), a small *i* button allows the user to display these dependencies for each Guidea. If a required idea is missing or an excluded Guidea is chosen, there will be a conflict in the model. The user can make choices, which lead to conflicts, but the application will draw the attention to the affected

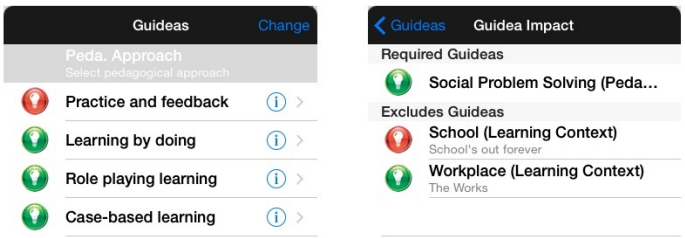

Figure 4.5: Menu to select a concrete option for the abstract Guidea "Peda. Approach" (left) and required/excluded Guideas for one of these options (right).

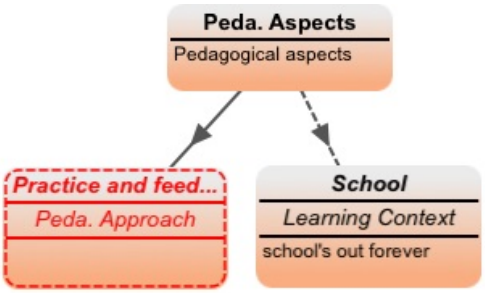

Figure 4.6: The "pedagogical apporach - practice and feedback" excludes the feature "school" as "learning context". This conflict is indicated by coloring the Guidea red.

Guidea. A small green or red light bulb beside the name of the Guidea indicates whether there would be conflicts with the current model, if the Guidea was selected. When a Guidea is selected despite an existing conflict, it is colored red (see figure 4.6).

4.2.3 Other Features

The application has other features, which are less relevant for the collaborative version. They shall be mentioned here for the sake of completeness.

Adding GuideaTemplates

GuideaMaps allows the user to add new meta models (called GuideaTemplates). When installing the application, a new tab in the iOS "Settings" App is created, where the user can enter the url of a GuideaTemplate. When the application is started the GuideaTemplate is added and can be selected

as starting point (meta model) when creating a new GuideaMap. In this way, different meta models can be made available for different software domains.

Sharing GuideaMaps

Application models created with GuideaMaps can be shared via email. The application creates a textual version of the current GuideaMap (i.e. application model), adds the xml version as attachment, and sends the message to an email address specified by the user. In this way, users can exchange application models, but the feature is also used to obtain a textual description of the application model.

Auto-Save

The application keeps track of changes made in the application model by automatically saving the document in the background. When the user closes the application, the current document is saved. When GuideaMaps is started again, the same document is opened automatically.

4.3 GuideaMaps - Implementation

This section provides a short overview of the implementation of GuideaMaps.

4.3.1 High Level Structure

The overall structure of the application follows a Model-View-Controller design pattern. The central controller in the application is the MainViewController (see figure 4.7). It coordinates interaction between the model (GuideaDocument and MetaModel) and the views. It manages a set of views, like GuideaPanel, which make up the user interface of the application. It re-sizes the layout of views, adjusts their content and responds to user interaction via the views. For handling Guideas and MetaGuideas the controller has references to the GuideaService and MetaService.

The MetaService is used to load a MetaModel from an XML file into memory. The GuideaService is responsible for loading the application model and maintaining it when the user is making changes to the model (e.g. selecting Guideas, adding comments etc) while keeping references to the MetaModel. The application model is visualized in the GuideaPanel view.

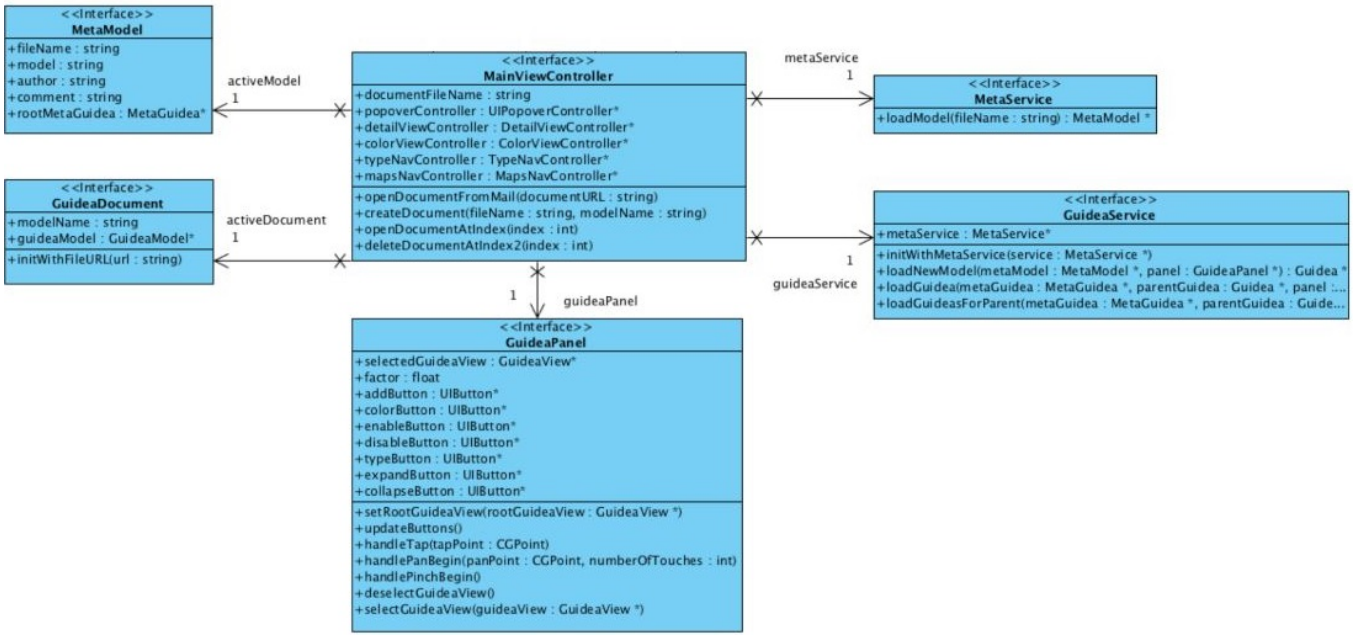

Figure 4.7: Class diagram of MainViewController (Janssens, 2013)

Beside the MainViewController, the application has a set of other view controllers for more specific tasks like DetailViewController (popover controller for setting Guidea details) and TypeTableController (table controller for showing available types of a Guidea).

4.3.2 Model and Meta Model

The application model and meta model are each represented by a tree, consisting of Guideas and MetaGuideas as nodes (see figure 4.8).

The meta model, representing the decision model, holds information about name, description, cardinality, abstract/derived boolean etc for each feature.

The application model, which consists of Guideas, stores information about the decisions taken by the user, like selected options for abstract Guideas or comments. Each Guidea holds a reference to its MetaGuidea. The x/y-coordinates of the Guidea on the screen are not stored in the Guidea object, but in a separate GuideaView object.

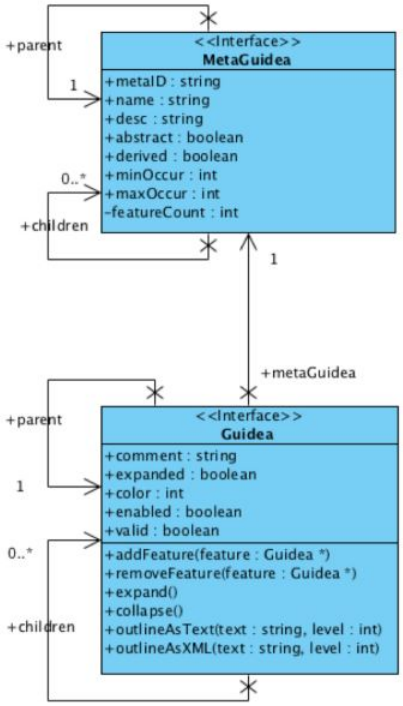

Figure 4.8: Class diagram of Guideas and MetaGuideas (Janssens, 2013)

The GuideaDocument (see figure 4.9) is responsible for loading files into the application and saving files. It holds a reference to the GuideaModel containing the application model via a reference to the root Guidea. The GuideaModel keeps a reference to the MetaModel, which holds the root MetaGuidea.

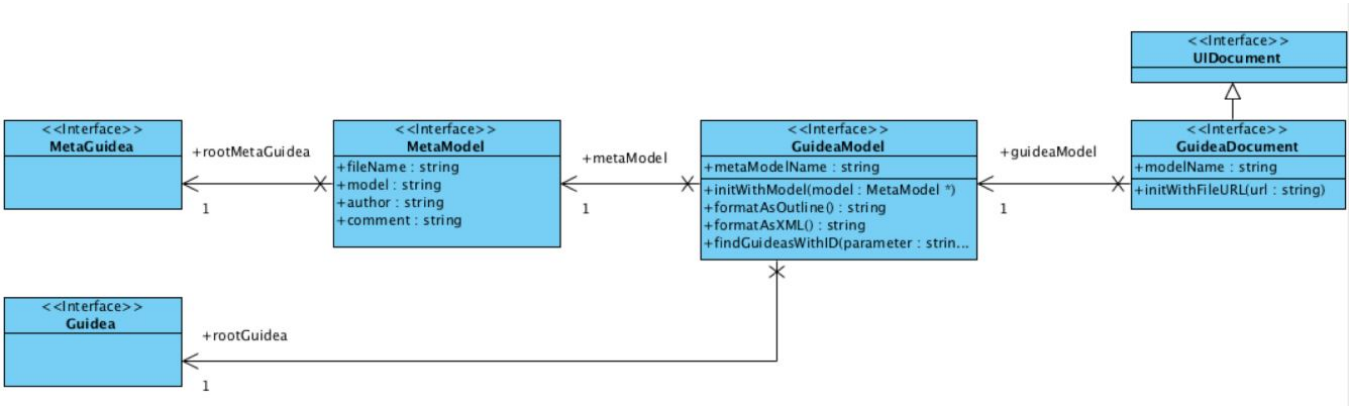

Figure 4.9: Class diagram of the model (Janssens, 2013)

4.3.3 Views

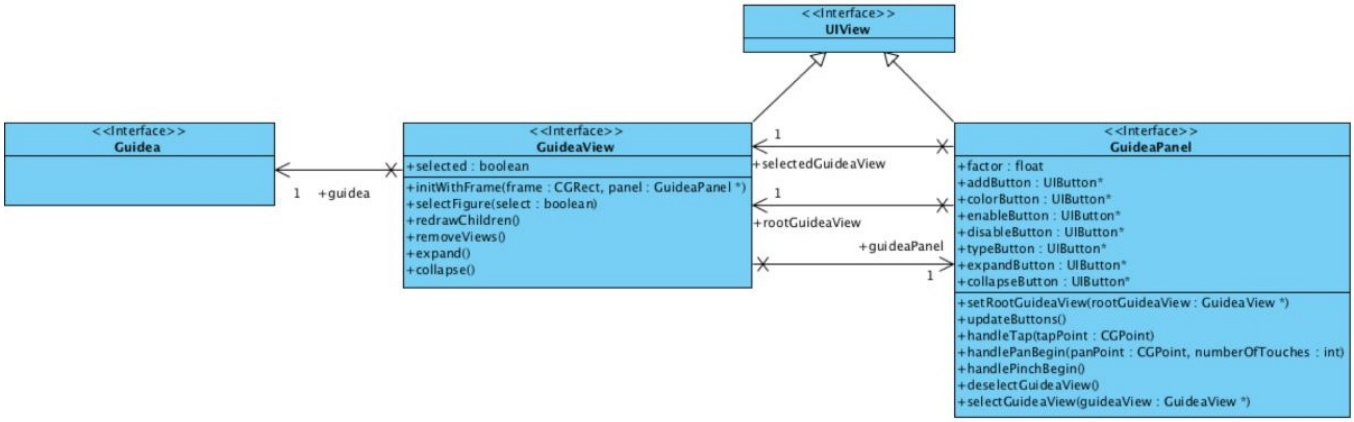

Figure 4.10: Class diagram Views (Janssens, 2013)

The GuideaView (see figure 4.10) is the visualization of a Guidea. It contains all functionality for drawing a Guidea and allows the user to select and deselect it.

The GuideaPanel is the visualization of the application model. It contains the GuideaViews, methods for user interaction like tapping, panning etc. and holds references to buttons for adding/deleting Guideas etc.

4.4 Summary

GuideaMaps uses concepts from feature modeling to support the user in the process of software requirement elicitation. Different types of domain specific requirements and their dependencies are defined in a decision model (GuideaTemplate) and stored as an XML document. Different GuideaTemplates can be created for different domains. The application visualizes the decision model as a hierarchical structure similar to a Mindmap. It provides an easy to use point, tap and drag user interface, allowing technical and non-technical stakeholders to document their decisions by selecting options and adding comments. The sum of those decisions is represented by a so called GuideaMap and stored as an XML document.

5

Collaborative GuideaMaps: Design

Requirement elicitation is a genuine group task, usually carried out as an iterative refinement process. This process involves individual work as well as group work. In meetings, requirements for different issues will be discussed and decisions will be taken. During individual work, stakeholders will prepare for these meetings and will usually concentrate on the requirements for their domain of expertise. To support this way of working, we propose to provide each participant with an iPad with GuideaMaps installed. When preparing for a meeting GuideaMaps is used as single-user application. During this phase the user goes through the map, selects options and adds comments to Guideas.

At the beginning of a group meeting one of the participants opens a common workspace, called GuideaSpace, and the other participants join it. Participants can go through the different Guideas one by one, see each others comments and discuss the advantages and disadvantages until the group agrees on a proposal. In that case, the decision is usually documented in the GuideaSpace, either by keeping the Guidea of one of the participants or by documenting the decision in the corresponding Guidea. If no agreement can be reached, the participants may decide to discuss another Guidea. After

the meeting the current state of the GuideaSpace is automatically saved on all devices. Now the user can work in private again and the process goes into the next iteration.

In order to support the collaborative process during group meetings, we introduce two main principles: the difference between private and public Guideas and the use of a common workspace called GuideaSpace. Public and private Guideas are discussed in section 5.1.1. The notion of public Guideas creates the need to design an appropriate visualization, which facilitates comparing content from different users. This visualization is presented in section 5.1.2. The role and functioning of the common workspace is discussed in section 5.1.3. As in a single-user GuideaMap, there may be conflicts between Guideas in the GuideaSpace. These conflicts can be between Guideas from different participants. However, as long as the group has not agreed on a final solution, they are only potential conflicts. This issue is discussed in more detail in section 5.1.4.

5.1 General Concepts and Design Principles

5.1.1 Public and Private Guideas

Since the application is meant to facilitate collaborative group work, the focus should lie on communicating and sharing ideas. However, if an idea was always visible for everyone, a simple blackboard would be sufficient. There are various reasons why the notion of a private workspace is essential:

- If the whole process of developing an idea, from the initial draft to a fully elaborated concept, was exposed to the group, the user could feel insecure. That would hinder the process of developing an idea. Periods of individual private work are an integral part of collaborative work and the tool has to support users to formulate their ideas in private.
- A participant might want to make annotations or comments just as a personal reminder and not share them with the group at all.
- A participant might want to present comments/ideas to the group in a certain order to make them more understandable or to support their viewpoint.

Introducing the concept of public and private workspaces to GuideaMaps (see requirement F3), while providing a fine-grained control over accessibility and visibility for the group, suggests the use of the notion of private and

public comments to Guideas. Since the application should be usable by casual users without technical knowledge or previous experience with the tool (see requirement U1), a complex rights management would over-complicate the process of sharing user content and add unnecessary complexity to the system. Currently, the content of a regular Guidea is limited to comments. There are two possible states in terms of visibility for a comment:

- private: only the local user (i.e. the owner) can see the comment
- public: every user in the GuideaSpace can see the comment

This state can be set or changed at any time, in single-user mode as well as in collaborative mode. In collaborative mode, a change is instantly visible for all users in the GuideaSpace (see requirement U2). When a new comment is added to a Guidea it is set to private by default. Visually, the current status of a comment is indicated by an icon at the bottom of the Guidea (see figure 5.1). The user can change between the public and private state by clicking the icon.

Figure 5.1: Comment on a Guidea with status public (left) and private (right)

5.1.2 Comparing Content from Different Users

The different stakeholders are meeting in order to present their opinions about the requirements for certain aspects of the software to the group. Usually the contributions of the participants will be related to their field of expertise. Since these fields might overlap or participants might want to comment on other fields as well, there will usually be several different comments for a particular Guidea. Content from different participants could be grouped by participant or by topic (i.e. Guidea). A central aspect of finding a solution collaboratively is the process of comparing different contributions. The group uses the GuideaMap as a thread and goes through the hierarchical structure comparing the different contributions for each Guidea. This process of guided ideation naturally demands grouping by Guidea. Therefore, in a Guidea all available comments will be shown (see figure 5.2) making it

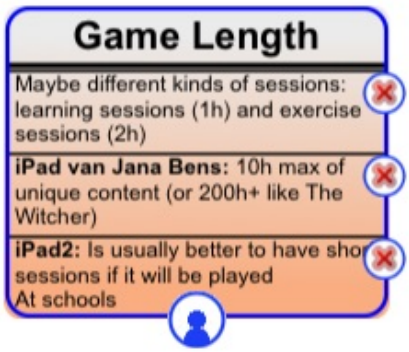

Figure 5.2: Comments from different users to one Guidea. The comment from the local user is always shown on top. Comments from other users are shown beneath with their user name added.

easy to compare contributions from different users (see requirement F4).

The comment of the local user will always be shown as the first comment of a Guidea, as shown in figure 5.2. The other participants are identified (see requirement U3) and their comments are shown beneath.

On the right side of each comment in a Guidea there is an x-button which allows the user to delete the comment. If a comment is deleted by its author, the comment is automatically discarded in the GuideaSpace, which means it disappears from all devices. It is also possible for a participant to delete (public) comments from other users. In that case the comment is deleted from the GuideaSpace, but it remains as a private comment in the Guidea on the device of the author. This is a precautionary measure to prevent users from irretrievably deleting other people's ideas. Even if a comment is deleted from the GuideaSpace, the author might still want to keep it and work further on it or reformulate it for the next meeting. Therefore the final decision of completely deleting a comment always lies with its author.

5.1.3 GuideaSpace - A Shared Workspace

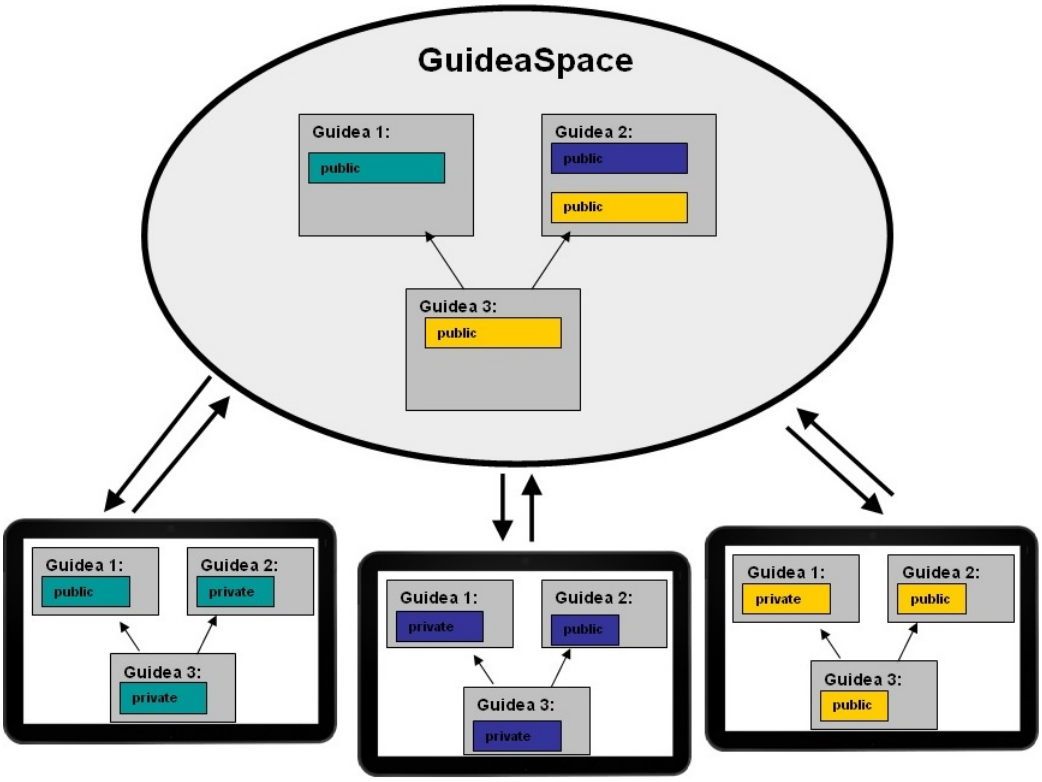

Figure 5.3: GuideaSpace with comments from three different users. Each user has comments to all three Guideas on the screen, but only the public ones are in the GuideaSpace.

As already explained, we have opted for the use of a public and a private workspace in GuideaMaps. Conceptually, the private workspace containing all private comments, and the public workspace containing all public comments, are separate and independent from each other. Figure 5.3 illustrates a GuideaSpace shared by three users. Each user has his own comments for the Guideas, but only the public ones are in the GuideaSpace.

As figure 5.2 shows, in the visualization of a GuideaMap these two conceptually separated spaces are combined in one view. A private comment is shown next to public comments, which allows the user to easily compare all contributions, whether he shared his comment or not.

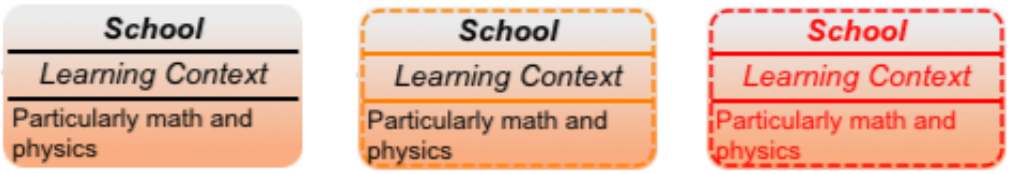

Figure 5.4: Guidea showing the three possible conflict states: no conflict (left), potential conflict (middle), conflict (right)

5.1.4 Conflicting Guideas

When several users work on a GuideaMap it can be hard to keep track of the different contributions in terms of coherence. The final outcome must be a GuideaMap that is consistent, i.e. where there are no conflicting Guideas. Creating a consistent GuideaMap can already be a tricky task for a single user. When different users are formulating their own requirements and try to merge their ideas to a uniform proposal, this becomes even harder. In order to support the users, the application automatically analyses the GuideaMap and points out whether combinations of Guideas cause conflicts (see requirement F5). Conflicts are indicated by coloring the Guideas involved. GuideaMaps distinguishes between three different conflict states for a Guidea (see figure 5.4):

- no conflict (black)
- potential conflict (orange): This refers to conflicts emerging from the collaboration of users. It is not yet a conflict, but demands the group's attention and usually requires the group to discard at least one contribution from a participant in order to resolve the conflict.
- conflict (red): This refers to a conflict in the model of a single user.

There are various situations that could lead to conflicts, especially in connection with public contributions from different users. We describe them in the next sub sections.

Cardinality Conflicts

As mentioned in section 4.2.2 cardinalities always appear in combination with abstract and derived Guideas. A cardinality indicates that there is a maximum for the number of children that can be selected at a time for an abstract Guidea. This limit can be smaller than the overall number of children.

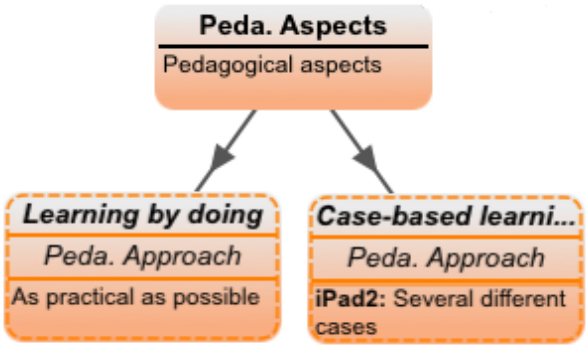

Figure 5.5: The maximum number of selected children (here: one) is exceeded, because different users have selected different "Pedagogical Approaches". The coloring indicates a potential conflict.

Let us assume there is an abstract Guidea with five children A,B,C,D,E, which has a maximal allowed number of selected children of three. User 1 can select A,B,C and user 2 can select C,D,E. The model of user 1 and the model of user 2, each taken individually, are not exceeding the limit. But if both users make their choices public, the GuideaSpace contains children A,B,C,D,E, which exceeds the maximal number of selected children. In that case all five children are viewed, but their frames are colored orange indicating a potential conflict. Now the group needs to decide which of the five options should be discarded in order to comply with the abstract Guideas' cardinality.

Dependency Conflicts

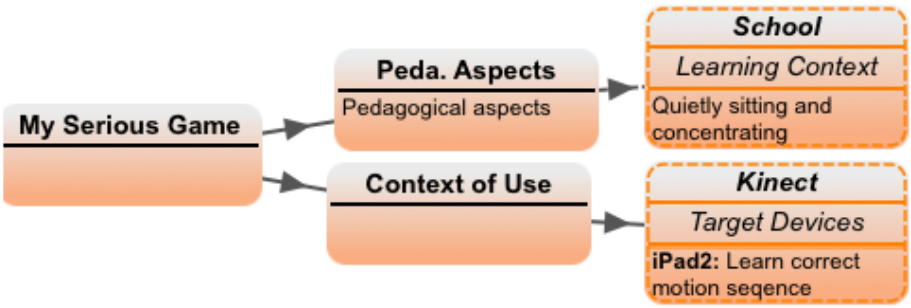

Figure 5.6: Dependency conflict between two Guideas

As explained in section 4.2.2 GuideaMaps can represent two different kinds

of dependencies between Guideas - a Guidea can "require" or "exclude" another Guidea. In single user mode, there is clearly a conflict in the model if one of these dependencies is violated. If two users have developed two distinct, consistent GuideMaps and both are added to the GuideaSpace, inconsistencies between Guideas from different users can appear.

Let us assume the decision model specifies that "Learning Context School" and the "Target Device Kinect" exclude each other. Figure 5.6 shows a situation where these two Guideas have each been added by two different users. Analyzed separately, both users have consistent models, but the GuideaSpace containing Guideas from both models results in a potential conflict.

This kind of conflict is only a potential conflict, but since the main goal is to combine ideas from the different participants into one consistent proposal, it is helpful to draw the user's attention to those potential conflicts.

5.2 User Interface

The user interface of GuideaMaps contains a menu bar on top of the screen (see figure 5.7). General functions are on the left side and functions relating to collaboration are located on the right side. Buttons and their functionality from left to right (p = existed in previous version):

- Create new GuideaMap or open existing GuideaMap (p)
- Reset GuideaMap (p)
- Create a new GuideaSpace or connect to an existing GuideaSpace
- Synchronize GuideaMap with GuideaSpace
- View GuideaSpace containing all public comments (no private comments)
- Send email with GuideaMap (p)

The functionality of those buttons and other aspects of the user interface are explained in more detail in the following sections.

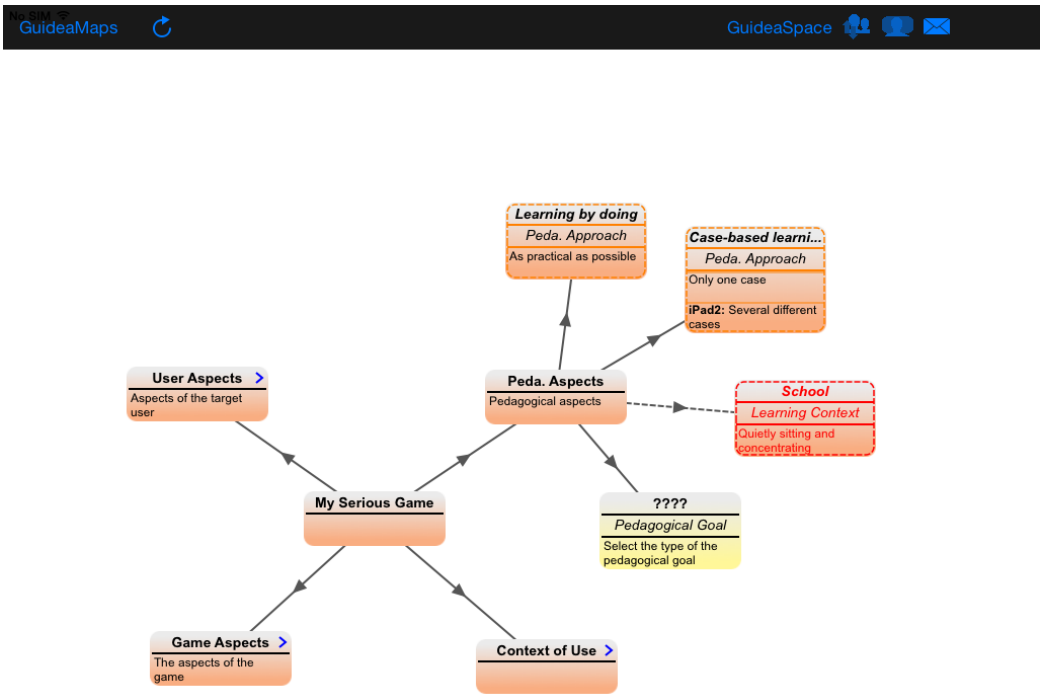

Figure 5.7: User interface of GuideaMaps with menu bar on the top. General functions are on the left and collaborative functions are on the right hand side. Buttons from left to right: Open GuideaMap, reset GuideaMap, connect via GuideaSpace, synchronize GuideaMap, view GuideaSpace, send email

5.2.1 Connecting to a GuideaSpace

In order to create or connect to a GuideaSpace, the user should select "GuideaSpace" on top of the screen (see figure 5.8). The popup menu allows the user to enter his name within the GuideaSpace and to open or join GuideaSpaces. At the beginning of a meeting one of the participants should activate the "Open GuideaSpace?" switch in order to open a GuideaSpace. All other participants can join this workspace by using the "Browse for GuideaSpaces" button (see figure 5.8).

Users can join a group at any time (not only at the beginning of a meeting) and synchronize their devices with the GuideaSpace (see requirement F10).

Once the group has established a GuideaSpace, one of the participants should

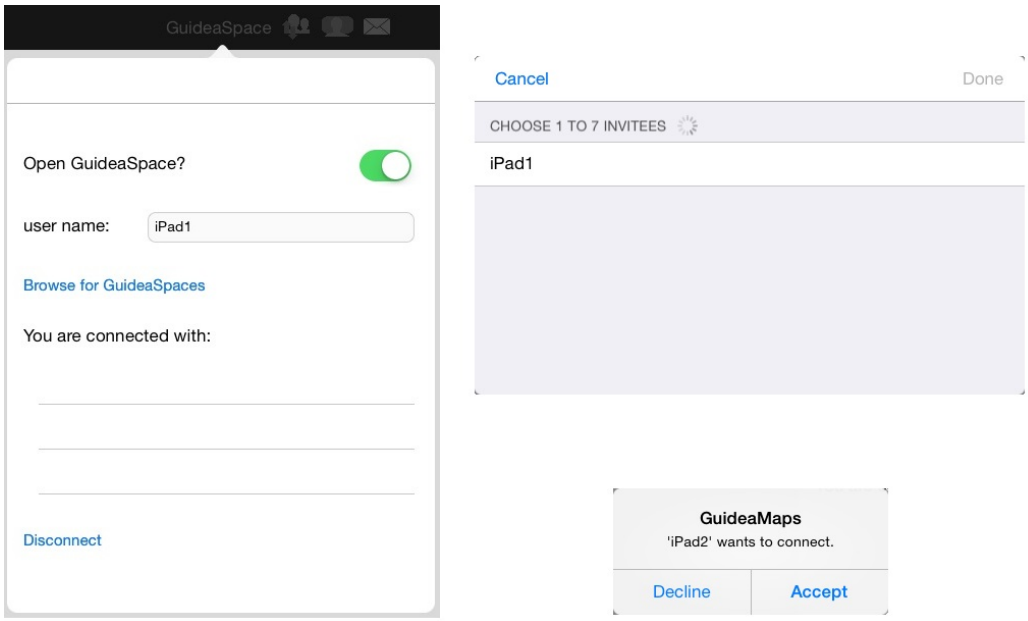

Figure 5.8: Menu for connecting in a GuideaSpace (left). Browsing for GuideaSpaces (right top). Accepting or rejecting request to join GuideaSpace (right bottom).

initiate the synchronization process. After the synchronization is finished, all devices are updated to the current state of the GuideaSpace. Synchronization is initiated by clicking on the button in the top bar.

5.2.2 Changing the GuideaMap in a GuideaSpace

A GuideaSpace itself is simply a shared workspace, which is initially empty. All users have to agree on which GuideaMap they want to work with and open it. But a GuideaSpace is not limited to one specific GuideaMap. Even after all users are connected, it is still possible to change the GuideaMap by clicking on *GuideaMaps* in the menu bar and selecting a Map. If users are working with different GuideaMaps and try to synchronize, an error message appears (see figure 5.9).

It is possible for the user to open other GuideaMaps while connected to a GuideaSpace. Lets assume a group of users have connected their devices in a GuideaSpace, try to synchronize and then discover that they are using different GuideaMaps (e.g. different versions of a GuideaTemplate). The users can look through their saved GuideaMaps until all of them have found

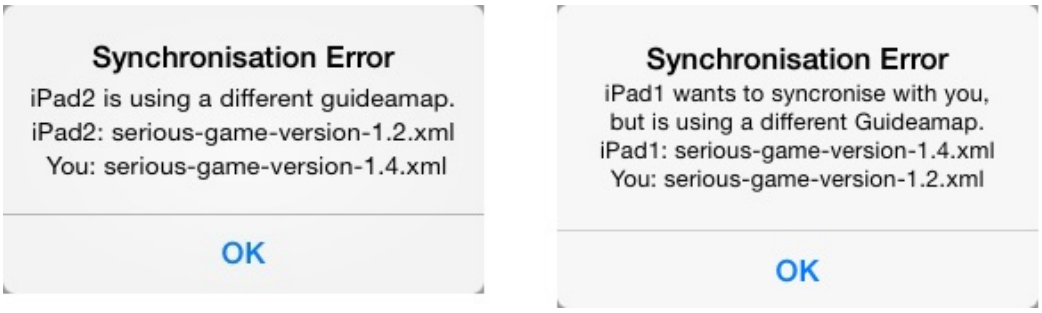

Figure 5.9: If users in the GuideaSpace are using different GuideaMaps and one tries to synchronize, both users are warned that the synchronization failed.

the right Map. Users can open different GuideaMaps without the need to disconnect and reconnect. This is handy because:

- The process of connecting devices can take a while especially in bigger groups.
- The creator of the GuideaSpace has to agree each time a user wants to connect.

If one of the users is changing GuideaMap while connected, the GuideaSpace is not affected. Public Guideas of the user stay in the shared work space.

5.2.3 End Session

At the end of a meeting, the users can simply disconnect by clicking on the *Disconnect* button in the *GuideaSpace* menu (see figure 5.8). The current state of the GuideaSpace including all public Guideas will be saved on all devices. When a participant continues to work alone, all public comments from the other participants are still available and can be used to follow up the last session and prepare for the next meeting (see requirement F6).
If a user decides to leave the meeting earlier, he can simply disconnect any time and the current state of the workspace is saved on his device (see requirement F9).

5.2.4 View GuideaSpace

As mentioned all public Guideas are visible on every device. However, there are different reasons why it can be useful to have a visualization of the GuideaSpace itself on a separate big screen or projected on a wall.

- Since private and public comments are shown in the same view on a single device, there is no clear separation between the private and the shared workspace.
- An additional screen showing only the GuideaSpace could help the participants to keep track of what the group has agreed on, which part of the map the group is currently working on and what still needs to be done.
- If some participants in the meeting do not have a device with GuideaMaps (or don't want to use one), they can still follow the discussion and even participate by looking at the GuideaSpace on the big screen.

One of the devices of the participants could be connected to a projector or big screen, but depending on the user and his way of using private comments, his GuideaMap might be cluttered with annotations which are not meant to be read by the group or are simply not interesting for the other participants. Therefore, simply projecting one participant's screen would not be helpful or may even be more confusing for the group.

For the purpose of projecting only the GuideaSpace, we provide a functionality that allows the hiding of all private Guideas. Only Guideas made public in the GuideaSpace are shown (see requirement F8). This is done by clicking on the "hide private content" button  in the top bar. The button will change to  ("show private content"). When clicking again the private comments reappear and the button again changes to the "hide private content" button.

5.3 Summary

GuideaMaps can be used by IT-professionals as well as non-technical experts (U1). The application supports a single-user mode, e.g. when preparing for a meeting (F1), as well as a collaborative mode. During a meeting participants can share their ideas in a GuideaSpace (F2). Users can develop ideas in private, but also share them with the group at at any time (F3). All changes to the shared workspace are instantly visible to all group members (U2). All comments on one Guidea are viewed next to each other, which facilitates the comparison of different users' contributions (F4, U3). The application supports users to keep track of dependencies in the model and points out different types of conflicts (F5). It is suitable for a longer process of requirement elicitation including phases of individual work and group work using

one GuideaMap (F7). Users are able to join and leave group meetings at anytime (F9). When leaving a session, the latest state of the shared work space is saved on all devices and accessible during phases of individual work (F6). The application can view a visualization of the current state of the GuideaSpace (F8). This visualization helps users to keep track of what the group has decided already and which topics still have to be addressed.

6 Collaborative GuideaMaps: Implementation

The original GuideaMaps application was developed as an iOS application. The goal was to extend this application with functionality to support collaborative use. The first part of this chapter gives a deeper insight into the development and implementation of iOS applications in general, i.e. the development environment, the programming language and the frameworks. The second part covers the technical aspects of the implementation of our extension, like classes, network communication etc.

6.1 General Aspects of iOS Development

6.1.1 Development Environment

The mobile operating system iOS is running on iPod, iPhone and iPad. Apple offers a modern developing environment for those devices [1]

- Xcode: Integrated development environment (IDE) including source editor and graphical user interface editor

[1] `https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/`

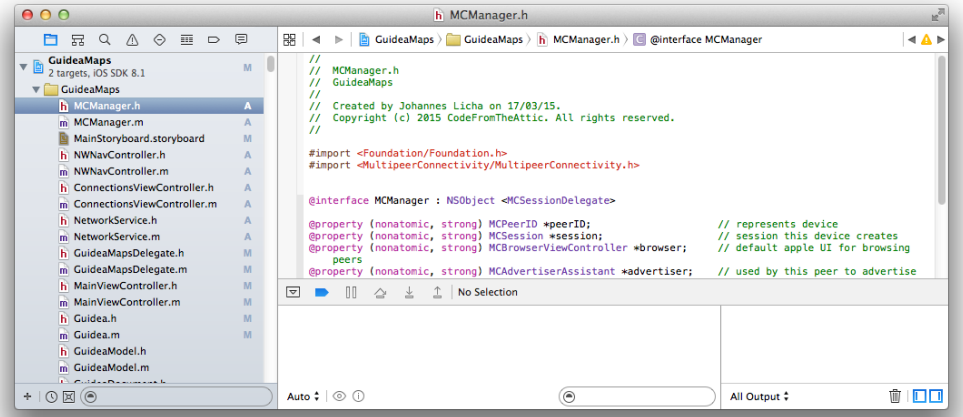

Figure 6.1: Developing in Xcode

- iOS SDK: extension for Xcode containing specific tools, compilers and frameworks for iOS development

These tools are required for App development on iPads and can be downloaded for free from Apple's App Store. In order to test applications and distribute them, enrollment as an Apple Developer in the iOS Developer Program is necessary.

Xcode contains all necessary features and combines different development tools. The user can browse through projects and edit classes, interfaces etc. It offers the functionality of designing user interfaces by adding elements and binding them to methods or classes via drag-and-drop. Xcode can run applications by simulating different types of mobile devices or deploy apps to registered mobile devices for testing purposes. A range of debugging functionalities like breakpoints, step-by-step execution etc. are available.

6.1.2 Programming Language

Until 2014 Objective-C [2] was used as the primary language for OS X and iOS development. It is a super set of C, so primitive types (e.g. int, double), flow control statements (e.g. if, while) and standard libraries (e.g. stdio.h) from C are supported. But it has been extended by (mostly object oriented) concepts like

[2] `https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html`

- classes
- interfaces
- static variables and methods
- instance variables and methods
- properties (with automatic synthesizing of accessors)
- single class inheritance
- multiple inheritance of specification (with protocols)
- dynamic typing

In 2014 Apple introduced the successor of Objective-C called Swift. However, since there is a large code basis in Objective-C available for this project and it is still widely used by other developers, Swift was not used for the prototype tool.

6.1.3 Frameworks

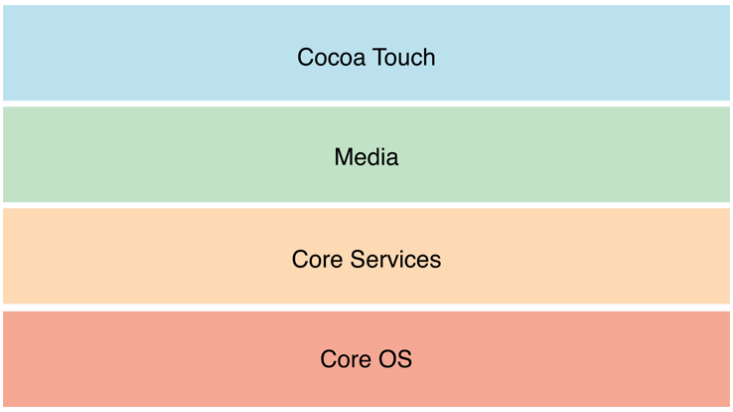

Figure 6.2: iOS layers

Creating applications for iOS does not mean developing everything from scratch, but rather using the right tools suitable for accomplishing the desired functionality. Apple provides many frameworks[3] for all kinds of features

[3] `https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html`

ranging from Games, to Media, to Data Storage.

Applications don't talk to the hardware directly, but communicate with the operating system through a set of well defined layers (Figure 6.2). Lower layers contain basic services and functionalites. Higher layers are built on top of lower ones and provide more abstract, object oriented features. Each layer contains a number of frameworks. These frameworks allow the developer to reuse well-established code, which has proven to be efficient and safe in many other applications. It is preferable to use frameworks on higher levels if possible as they offer abstract concepts, hide complexity of lower layers and reduce the amount of code a programmer has to write.

GuideaMaps uses the following frameworks:

- Multipeer Connectivity - peer to peer connectivity
- UIKit - basic functionality for graphical, event-driven applications
- MessageUI - sending emails and other messages from within the application
- Foundation - collection data types, string management, threads etc
- CoreGraphics - native drawing engine supporting custom 2D-vector- and image-based rendering

The Multipeer Connectivity Framework[4] (MCF) will be covered in more detail in section 6.2.1, since it is not a standard framework used in every application and plays an essential role for realizing collaborative features in GuideaMaps.

6.1.4 Design Patterns

In software engineering, a design pattern is a general, reusable solution to a commonly occurring problem within a given context in software design[5]. Some design patterns, which have been used at different points in GuideaMaps are explained in the following sections.

```
@interface ConnectionsViewController : UIViewController <UITextFieldDelegate,
    MCBrowserViewControllerDelegate,UITableViewDelegate, UITableViewDataSource,UITextFieldDelegate>

@property (weak, nonatomic) IBOutlet UITextField *txtName;

- (void)textFieldDidEndEditing:(UITextField *)textField;

...

@end
```

```
@implementation ConnectionsViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    [_txtName setDelegate:self];
    ...

}

- (void)textFieldDidEndEditing:(UITextField *)textField{
    [NetworkService setNwName:textField.text];
}

...
```

Figure 6.3: Interface of the class ConnectionsViewController, which is conforming to the UITextFieldDelegate. The class implements the method textFieldDidEndEditing, which is defined in that protocol.

Delegation

An object which acts on behalf of, or in coordination with, another object, triggered by an event, is called a delegate[6]. The delegating class is often called responder. Usually this design pattern is used by frameworks. Figure 6.3 shows a simple example of how the pattern is used in GuideaMaps. The ConnectionsViewController acts as delegate for the text field txtName (name of the user in the GuideaSpace). After the text field is edited, the method textFieldDidEndEditing in the Controller is called.

Model-View-Controller

The Model-View-Controller design pattern distinguishes three different types of classes: Model, View, Controller. The model encapsulates application specific data and provides methods to manipulate and process it. A View displays data from a model and enables the user to edit data. The controller

[4] https://developer.apple.com/library/ios/documentation/MultipeerConnectivity/Reference/MultipeerConnectivityFramework

[5] https://en.wikipedia.org/wiki/Software_design_pattern

[6] https://developer.apple.com/library/mac/documentation/General/Conceptual/CocoaEncyclopedia/DelegatesandDataSources/DelegatesandDataSources.html

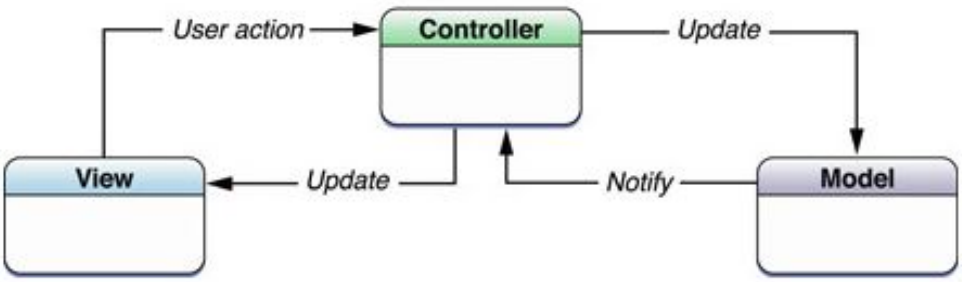

Figure 6.4: Illustration of the Model-View-Controller design pattern

acts as an intermediary between views and models. It updates views if the model changes and vice versa (see figure 6.4).

6.2 Architecture

The main structure of the original application, as presented in section 4.3, could be preserved during the process of adapting it to collaborative work. However, changes in most of the classes were necessary and the architecture had to be extended to be able to connect devices and enable network communication. Those changes and extensions are presented in the next sections.

6.2.1 Network Architecture

The application does not use an external server to keep the decision and application model. There are two main reasons for this design choice:

- The application is not computationally intensive and the amount of data to be stored is very limited. An application or database server is not necessary.
- Even though many meeting rooms are equipped with WiFi, relying on a network or even internet connection for communication with the server is an unnecessary limitation (see requirement F9).

In order to conduct a meeting, the participants only need to bring their devices (iPads) with GuideaMaps installed.

All devices hold a complete version of the application model. If one device played a dedicated role (e.g. centrally storing the model), the group would depend on that device to hold a meeting, which is not practical. GuideaMaps is meant to be used in a number of consecutive meetings. If one or more users are prevented from attending, the group must still be able to conduct a meeting.

During a meeting one device acts as a server. From the users' perspective that device does not play a special role apart from opening the GuideaSpace. On a technical level it provides the service "GuideaSpace", creates the session and invites other peers to join it (see section Multipeer Connectivity Framework). However, the role as a server is restricted to network communication - as described above every device holds a complete version of the application model.

Multipeer Connectivity Framework

Apple provides a number of frameworks for connecting iOS devices[7]:

- CFNetwork provides a set of high-performance C-based interfaces that use object-oriented abstractions for working with network protocols. It is based on BSD sockets and supports various technologies like encrypted SSL or TSL connections, HTTP, FTP etc. This framework does contain all necessary technologies needed for our purpose. But since it's abstractions are rather low level (compared to e.g. MCF), using it would require a considerable effort.

- CloudKit provides functionality to connect an application to Apple's iCould. Even though this is a possible way to enable several devices working on one document (e.g. GuideaMap), it requires an internet connection, which is a serious limitation (see requirement F9).

- The Multipeer Connectivity Framework (MCF) is specifically meant for connecting nearby devices without requiring internet connectivity. MCF provides options for discovering and managing network services and makes it possible to create multipeer sessions with reliable in-order and real-time data transmission. It provides support for network connections over Wi-Fi, peer-to-peer Wi-Fi and Bluetooth to connect up

[7] `https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.html#//apple_ref/doc/uid/TP40007898-CH10-SW14`

to eight devices. This framework provides all functionality necessary to develop a collaborative version of GuideaMaps with reasonable effort.

Therefore, the Multipeer Connectivity Framework was found to be the best match for our purpose. This section describes how devices running GuideaMaps are connected by means of the MCF.

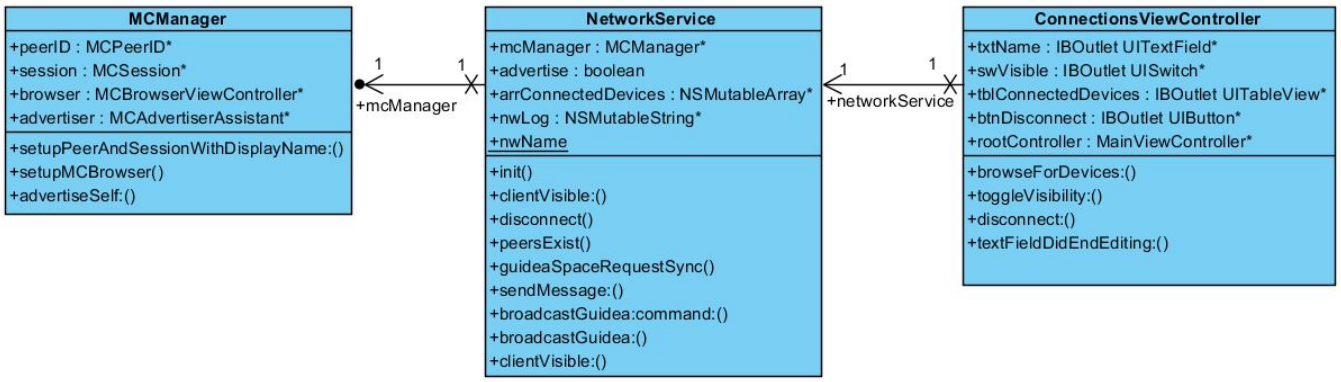

Figure 6.5: Class diagram: Classes used to connect devices in a GuideaSpace using the Multipeer Connectivity Framework.

The networking functionality of the application is provided by three classes (see figure 6.5): ConnectionsViewController, NetworkService, MCManager.

The ConnectionsViewController manages the popover menu allowing the user to open or connect to a GuideaSpace. It keeps references to fields and buttons and provides methods responding to user interactions like connect etc.

The NetworkService provides all functionality for network communication to other classes in the application. It can send requests, messages, commands and Guideas to other devices in the GuideaSpace.

The MCManager is responsible for the network connection on the level of the framework. It holds and manipulates objects like sessions, advertisers etc. provided by the MCF.

Each device is uniquely identified by a MCPeerID. The application advertises a service on the network using an MCAdvertiserAssistant. This service is an invitation to other devices to join a common MCSession. After joining a session, the framework supports exchange of message-based data and streaming data (e.g. serialized Guideas).

Network Communication

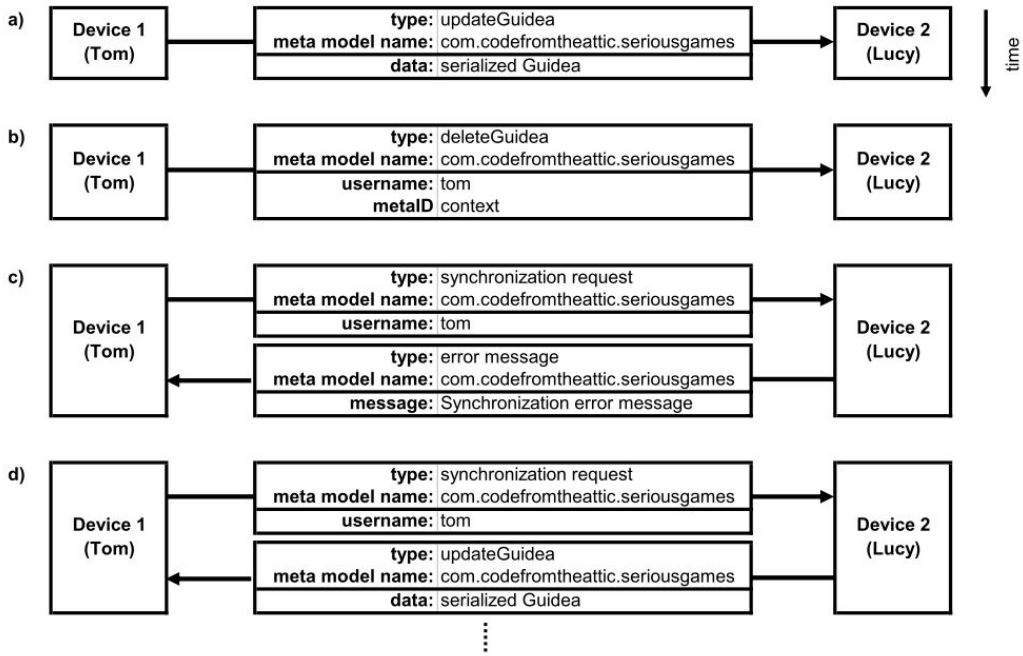

Figure 6.6: Examples of network communication in GuideaMaps: a) A serialized Guidea object, b) user has deleted Guidea, c) Failed synchronization, d) successful synchronization of GuideaSpace

The application supports different types of messages for communication between devices in the GuideaSpace:

- Guidea object
- command
- error message

All three types are composed of the following parts:

- type e.g. updateGuidea
- name of the meta model: to ensure that a user with the wrong GuideaTemplate cannot change the application model
- data

In figure 6.6 those three types are illustrated by typical examples of communication in a GuideaSpace (here only with two devices):

a) The user of device 1 has changed a Guidea e.g. changed a comment. A Guidea object with the changed data is serialized and broadcast to all devices in the GuideaSpace. Devices 2 updates his application model.

b) The user of device 1 has deleted a Guidea. A message with a delete-command containing the username of device 1 (tom) and the meta-ID of the Guidea is broadcast to all devices in the GuideaSpace. Device 2 finds the Guidea with the meta-ID in the application model and deletes the comment associated with the username tom.

c) Device 1 joined the GuideaSpace and clicks the synchronize button. A requestSync command is broadcast to all devices in the GuideaSpace. In this case Device 1 and Device 2 are using different GuideaTemplates. Device 2 is responding with an error message to the device which sent the synchronization request.

d) Device 1 sends a synchronization request. This time both devices are using the same GuideaTemplate and the synchronization is successful. Device 2 responds by sending back all Guideas with contributions from Lucy.

6.2.2 Application Model

The application model is represented by a tree of Guidea objects. Each Guidea holds the following information about all users in the GuideaSpace:

- username (string)
- comment (string)
- is the comment public? (boolean)
- is the Guidea enabled? (boolean)

Each Guidea keeps information about its conflict state:

- Conflicts between Guideas of the local user (valid)
- Conflicts between Guideas from different users (potentialConflicts)
- Is the number of allowed occurrences of an abstract Guidea exceeded (maxOccurExceeded)

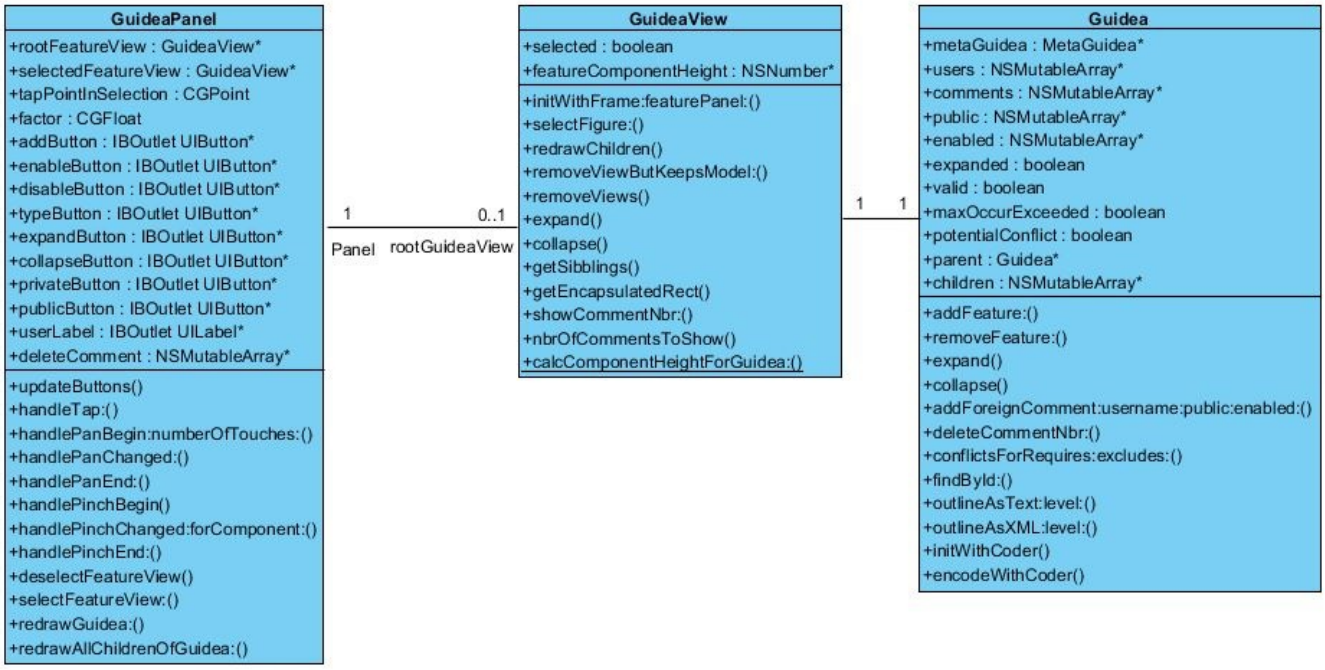

Figure 6.7: Class diagram of Guidea with GuideaView in a the panel

The Guidea class conforms to the NSCoding protocol. That enables Guidea objects to be (de)serialized, for the purpose of network communication, using (de)coders provided by the Foundation framework.

The visual representation of a Guidea is a GuideaView. It provides methods for drawing itself, but also updating/resizing since the size depends on the number of user comments.

All GuideaViews are drawn in a GuideaPanel. This panel has references to all buttons (make Guideas public, delete comments etc.). It provides methods for handling user interactions and drawing buttons. The set of buttons shown depends on the type and state of the selected Guidea and the number of comments.

Our modification of the structure of application models in GuideaMaps to contain contributions from different users entailed a number of necessary changes and extensions in the following parts:

- Methods for analyzing the model to find conflicts

- The XML-parser, which reads the saved document and creates the tree of Guidea objects in memory
- The methods for XML serialization of Guideas
- The XML Schema for the application model (see section 6.2.3)

6.2.3 XML Schema for the Application Model

An XML schema defines a type of XML document through constraints on structure and content. The following XML Schema Definition (XSD) contains the structure of the collaborative GuideaMaps application model. It is a variation of the XSD in section 4.1.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="
    qualified">
    <xs:complexType name="guideaType">
        <xs:sequence>
            <xs:element name="comment" type="commentType" minOccurs="0"
                maxOccurs="unbounded"/>
            <xs:element name="guidea" type="guideaType" minOccurs="0"
                maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="xPos" type="xs:float" use="required"/>
        <xs:attribute name="yPos" type="xs:float" use="required"/>
        <xs:attribute name="expanded" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="commentType" mixed="true">
        <xs:attribute name="username" type="xs:string" use="required"/>
        <xs:attribute name="enabled" type="xs:string" use="required"/>
        <xs:attribute name="public" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:element name="guideaModel">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="guidea" type="guideaType"/>
            </xs:sequence>
            <xs:attribute name="metaModel" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

A guidea has a unique identifier, the x/y-position on the screen and an indicator whether its children are currently shown or not (expanded). It can contain other Guideas (its children) and comments from different users.

Since a Guidea not only shows the comment from the local user, but potentially also comments from all users in the GuideaSpace, the commentType contains a username and two indicators, describing if that user has enabled

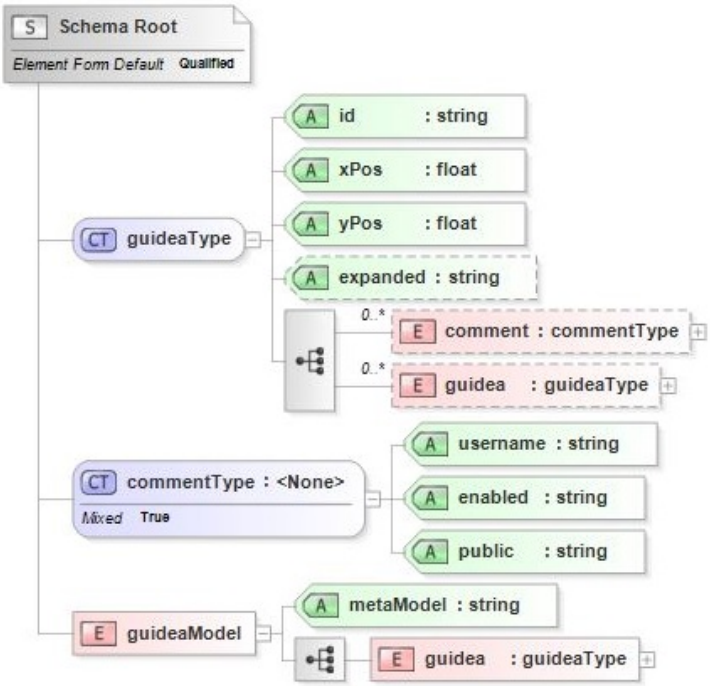

Figure 6.8: XML Schema Definition of the application model

the Guidea on his device and if it is currently visible for other users (public).

The application model itself is composed of the name of the meta model (GuideaTemplate) and the root Guidea containing the whole tree structure of the model as its children.

The following code is an example of an application model valid with respect to the given XML Schema Definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<guideaModel metaModel="com.codefromtheattic.games.xml">
    <guidea id="root" xPos="10" yPos="10" expanded="TRUE">
        <guidea id="paspects" xPos="360" yPos="166" expanded="TRUE">
            <comment username="Tom" enabled="TRUE" public="TRUE">set focus
                more on education</comment>
            <comment username="Anna" enabled="TRUE" public="TRUE">its
                crucial to keep user motivated</comment>
            <comment username="Alice" enabled="TRUE" public="TRUE">involve
                user in decisions</comment>
            <guidea id="approach" xPos="140" yPos="274">
                <comment username="Tom" enabled="TRUE" public="TRUE">connect
                    children from different countries</comment>
```

```
                <comment username="Alice" enabled="TRUE" public="TRUE">
                    consider users age</comment>
            </guidea>
            <guidea id="context1" xPos="540" yPos="51">
                <comment username="Tom" enabled="TRUE" public="FALSE">
                    connect children from different countries</comment>
            </guidea>
        </guidea>
    </guidea>
</guideaModel>
```

6.3 Summary

Our extension of the original GuideaMaps is an iOS application developed in objective C. It uses the Multipeer Connectivity Framework to connect up to 8 devices via Bluetooth. The application does not rely on an external server (see requirement F9). Every device holds a complete version of the application model, which enables all users to open a GuideaSpace.

The tool is a modified and extended version of the GuideaMaps application (see chapter 4). The adaption of the application model entailed many other modifications throughout the project. The visualization of the application model, XML parsing, XML serialization and XML Schema had to be modified. New types of conflicts between Guideas demanded the development of application logic to analyze the model. The implementation of the networking feature includes the sending, receiving and processing of commands, Guideas and error messages.

7
Evaluation and Future Works

This chapter describes the evaluation of Collaborative GuideaMaps and its outcome. In section 7.1 the procedure of the evaluation is described in detail. Section 7.2 outlines some ideas for future works.

7.1 Evaluation

The evaluation of the tool was conducted as an informal case study (Lazar, Feng, & Hochheiser, 2010). The application was used during a meeting to elicit the requirements for a serious game for youngsters to learn some aspects of proposition logic. The goal of this case study was to obtain initial feedback on the usability of the tool and to find points of improvement. The participants of the meeting were members from the department of computer science at VUB (Prof. Dr. Olga De Troyer and four Phd students):

- Teacher of a course in logic (and client)
- A pedagogically schooled person
- The main developer of the game (CS schooled, knowledgeable about games)
- Two game developers

Setting:

1. The participants are informed about the goal of the case study: Develop an initial set of requirements for a serious game for youngsters to learn some aspects of proposition logic.

2. Each participant receives an iPad with the tool installed.

3. The participants get a short introduction to the tool.

4. The participants prepare some ideas individually for 30 minutes. They are encouraged to focus on their domain of expertise, but can also contribute in other fields.

5. A meeting is held, where the participants should agree on the initial requirements for the game. The main developer for the game acts as chair. The meeting is stopped after 60 min.

6. The participants give feedback on the application.

The participants are observed the whole time. They are encouraged to think out loud and give direct feedback.

Some bugs in the application were detected (most of them were already known):

- If users add very long comments to a Guidea, the box is not big enough and part of the text is cut off

- Sometimes Guideas are overlapping. This could be fixed by automatically rearranging the map when new Guideas are added

- The user can only zoom in and out to a certain degree. This has been perceived as an unnecessary limitation.

- The process of finding and connecting to other devices is slow and sometimes fails, if there are more than two devices in the GuideaSpace. This problem wasn't discovered until the evaluation, because there were only two devices available during the development of the tool.

One interesting discovery unveiled by the evaluation concerns a basic characteristic of the application. The fact that the tool is comment-based led to some confusion among the participants. This issue is discussed in section 7.2.2 together with possible solutions.

7.2 Improvements and Future Work

7.2.1 General Improvements

Some issues with the current version of the tool have been raised and the users came up with interesting ideas how the application could be improved. The participants also proposed some additional features for future versions.

- All buttons are hidden if a Guidea is not selected by the user. The button which is used to change the state of a Guidea (public or private) is also used to show the current state. This state should be visible all the time, not just if the Guidea is selected. The users suggested to indicate the states with different colors.

- A Guidea can be enabled by one user and disabled by another user. This situation should be treated as a conflict and the Guidea should be colored to draw the user's attention to this inconsistency.

- Support the user in keeping track of the current state of the GuideaMap - which parts are finished already and which topics are still pending.

  - Every Guidea could hold an additional state: *pending* or *finished* It indicates whether the group still needs to address this issue or has discussed it already and came to a conclusion.

  - An additional comment section on the bottom of every Guidea could be used to capture the group's final decision. If this section is filled in, the Guidea is automatically set to *finished*.

- The group could be informed if one of the users left the GuideaSpace.

- The group could be informed if a Guidea is changed e.g. another comment is added. This happens regularly during a meeting and has to be done discretely e.g. the Guidea flashes once.

- It could be useful to have a button, which would make a Guidea and all its children, public.

- A color could be assigned to every user in the GuideaSpace. This color could be used e.g. to indicate the author of a comment.

7.2.2 From a Comment-Based to a Selection-Based System

The collaborative features of the application are comment-based, i.e. users fill in comments to express their requirements. Comments can be shared with other participants. However, during the evaluation, in some cases users just selected an option (a derived Guidea) without adding a comment. This selection is not visible for other users.

One possible solution could be to automatically open the comment popover menu after selecting an option and force the user to add a comment. However, in some cases it makes sense to select an option without adding a comment. If the user has to make a simple choice, it is not practical to demand a comment for every selected option.

A more realistic solution would be a selection-based approach for derived Guideas. Irrespective of whether a comment has been added or not, a derived Guidea would have the private/public button. If the derived Guidea was made public, it would be added to the GuideaSpace and made visible to other users.

7.2.3 A History for GuideaMaps

A history is an interesting feature for single-user applications and collaborative tools. In a group work situation where different opinions and ideas are weighed against each other, the ability to undo and redo changes becomes particularly useful. Therefore, our new application would certainly benefit from this feature. This section contains a proposal for the design of a history.

The history is visualized as a time line, located at the bottom of the screen (see figure 7.1). Each change in the GuideaMap is represented by a dot. The user can scroll back and forth through the history by moving the time line to the left and right via drag and drop. The state of the GuideaMap which is currently being viewed is represented by the slightly larger dot in the middle of the time line. While navigating through the history, the screen is always showing the state of the GuideaMap represented by the slightly larger dot in the middle. The user can click on a dot to see a small description of the change, which led to the state it represents.

GuideaMaps allows the user to hide private Guideas (see section 5.2.4). If only public Guideas are shown, the history will also only contain changes to

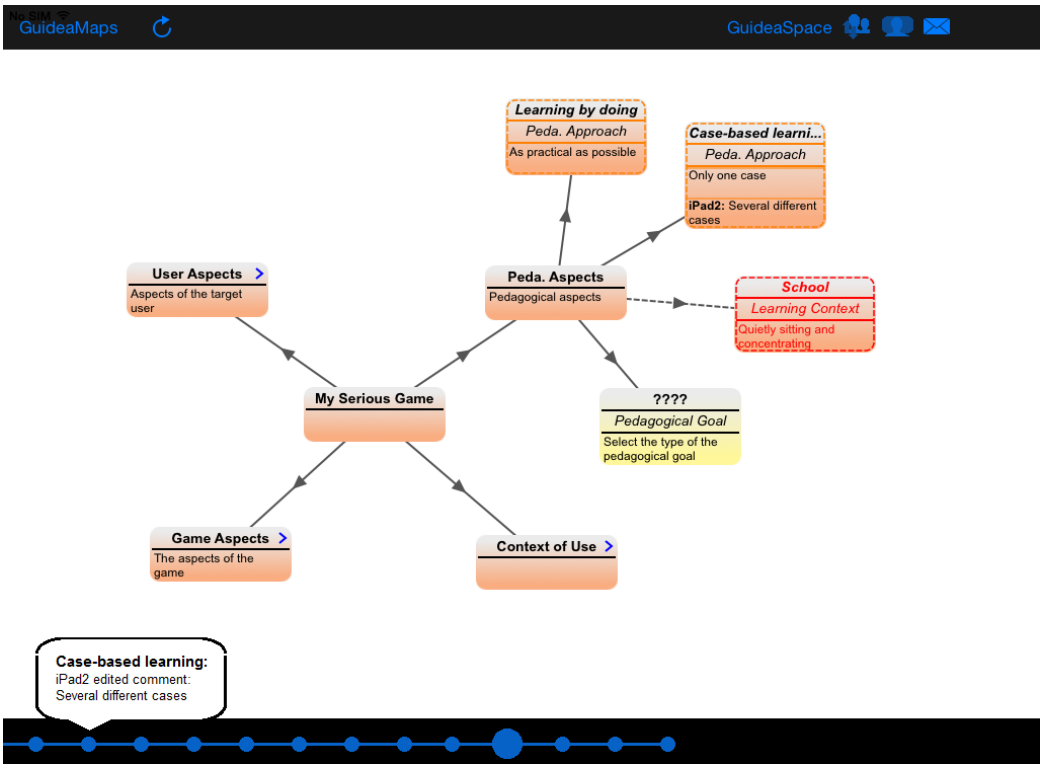

Figure 7.1: Design of user interface for history

public Guideas.

7.3 Summary

The evaluation of the application was conducted as an informal case study where the tool was used during a meeting to elicit the requirements for a serious game. Some bugs were discovered and the users gave interesting suggestions for further improvements, such as changing the comment-based approach for derived Guideas to a selection-based approach. A history, allowing the users to undo and redo changes in the application model, is a feature that could be added in future versions.

8 Conclusion

Software requirement elicitation is a genuine group task usually involving technical and non-technical experts. The diversity of the participants' backgrounds can cause problems in meetings. It can be hard to find a common language to discuss interdisciplinary issues. Notations for the purpose of capturing ideas, problems and solutions are often too technical to be understood by all participants. Inefficient communication between stakeholders is a common problem, impeding progress in meetings and hindering the process of requirement elicitation.

In recent years tablets like iPads are commonly used in meetings. Exploring their potential to support users in meetings, in particular for domain specific software requirement elicitation, was the goal of this master thesis. It provides an answer to the research question "How can tablet software support technical and non-technical stakeholders when working collaboratively in the field of domain-specific software requirement elicitation?".

The Design Science Research Methodology was used as a method to investigate this question in a structured manner. After identifying the research problem and its motivations, the objectives for a solution were defined in the form of functional and usability requirements. A study of related scientific literature and available software revealed that no available application ful-

filled those requirements. That led to the development of a collaborative tool for domain-specific requirement elicitation. An existing single-user tool for iPad called GuideaMaps acted as a starting point. It uses a guided ideation approach and visualizes the users' decisions in a hierarchical structure similar to a mindmap.

The new application can be used in individual work and group work. It allows users to connect their devices in a meeting and share their ideas. It provides a private and a public workspace and supports the group in comparing their contributions and finding a common solution.

To obtain initial feedback on the usability of the tool, a pilot evaluation was carried out, conducted as an informal case study. It revealed problems and limitations of the application and inspired some general improvements and possible future works.

References

Al-Rawas, A., & Easterbrook, S. (1996). Communication Problems in Requirements Engineering: A Field Study. *Proceedings of the First Westminster Conference on Professional Awareness in Software Engineering, Royal Society, London*(February), 1–12.

Buzan, T. (2004). *The Mindmap Handbook*. HarperCollins.

Cechticky, V., Pasetti, a., Rohlik, O., & Schaufelberger, W. (2004). XML-Based Feature Modelling. *Software Reuse: Methods, Techniques, and Tools*, *Volume 310*, 101–114. doi: 10.1007/978-3-540-27799-6_9

De Troyer, O., & Janssens, E. (2014a). A Feature Modeling Approach for Domain-Specific Requirement Elicitation. *Requirements Patterns (RePa), 2014 IEEE 4th International Workshop on*, 17–24.

De Troyer, O., & Janssens, E. (2014b). Supporting the requirement analysis phase for the development of serious games for children. *International Journal of Child-Computer Interaction*, *2*(2), 76–84. Retrieved from `http://dx.doi.org/10.1016/j.ijcci.2014.05.001`

Geisser, M., & Hildenbrand, T. (2006). A method for collaborative requirements elicitation and decision-supported requirements analysis. *IFIP International Federation for Information Processing*, *219*, 108–122. doi: 10.1007/978-0-387-34831-5_9

Jaafar, J., Atan, M., & Nazatul, N. A. A. H. (2011). Collaborative Mind Map tool to facilitate Requirement Elicitation. *3rd International Conference on Computing and Informatics, ICOCI 2011*(067), 214–219. Retrieved from `http://www.icoci.cms.net.my/proceedings/2011/papers/214.pdf`

Janssens, E. (2013). *Supporting Requirement Elicitation for Serious Games using a Guided Ideation tool on iPad. Masterthesis.* VUB.

Kang, K. C., Cohen, S. G., Hess, J. a., Novak, W. E., & Peterson, a. S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study. *Distribution*, *17*(November), 161. Retrieved from `http://www.sei.cmu.edu/reports/90tr021.pdf` doi: 10.1080/10629360701306050

Lazar, J., Feng, H. J., & Hochheiser, H. (2010). *Resarch methods in human-computer interaction.*

Peffers, K. E. N., Tuunanen, T., Rothenberger, M. a., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. Retrieved from `http://search.ebscohost.com/login.aspx?direct=true\&db=bth\&AN=28843849\&site=ehost-live` doi: 10.2307/40398896

Prante, T., Magerkurth, C., & Streitz, N. (2002). Developing CSCW Tools for Idea Finding - Empirical Results and Implications for Design. *Cscw'02*(Cscw), 106–115. doi: 10.1145/587078.587094

Shih, P. C., Nguyen, D. H., Hirano, S. H., Redmiles, D. F., & Hayes, G. R. (2009). GroupMind : Supporting Idea Generation through a Collaborative Mind-mapping Tool. *Proceedings of the ACM 2009 international conference on Supporting group work*, 139–148. Retrieved from `http://portal.acm.org/citation.cfm?id=1531674.1531696` doi: 10.1145/1531674.1531696

Wieringa, R. (2009). DS as Nested Problem Solving.

Zarinah, M. K., & Siti Salwah, S. (2009). Supporting Collaborative Requirements Elicitation Using Focus Group Discussion Technique. *International Journal of Software Engineering and Its Applications*, *3*(3), 59–70.