Vrije
Universiteit
Brussel

Supporting Requirement Elicitation for Serious Games using a Guided Ideation tool on iPad

Master Thesis – Master Toegepaste Informatica

Academic Year 2012-2013

Erik Janssens

Promoter: Prof. Dr. Olga De Troyer

Table of Content

Acknowledgments

I would like to thank my promoter Prof. Dr. Olga De Troyer for her personal guidance, advice and support regarding this master thesis.

I would also like to thank my wife Annemie Schoofs and my kids Daan and Marlies for their love, patience and encouragement during my academic study and for supporting me getting my master degree.

I can image it must be a weird experience for them when their father decides to enroll for a master degree program when his kids are also studying at a university :-)

Thank you all.

Abstract

Requirement elicitation is one of the early phases in the development cycle for any software. Serious Games is no exception; a thorough problem analysis is needed before starting to design a Serious Game, in particular before defining the scenario for the game. However, currently there is no structured approach and few guidelines on how to tackle this phase. Developing games and also serious games is mainly considered as a highly creative process. Although, developing games and serious games requires a considerable amount of creativity, the development of those games could benefit from a more structured and guided development, for two main reasons: to reduce the development cost and to avoid failures.

The purpose of this thesis is to provide support for the requirement elicitation for serious games. As, in general, many different types of people (i.e. with different backgrounds) are involved in the development process of a serious game, we aim for a simple and easy to use tool. We therefore opted for a visual tool on a (iPad) tablet.

This thesis starts with an investigation on visual tools that are currently available on an iPad tablet that can guide a user who is not fully familiar with the problem domain of games definition. It then provides a proposal for capturing the decision model using a feature-based model and presents a Graphical User Interface (GUI) to assist the users in the requirements capturing.

In the second part, the thesis provides details on the implementation of the prototyped “GuideaMaps” application. This native iOS app runs on iPad devices and aids the user in collecting the required information and helps him with the decision-making process.

The visual tool gently forces different people (with different background) involved in the development of a Serious Game (e.g. against cyber bullying) to reflect on the goals, characteristics and main principles of a new to develop serious game. The tool can be used in meetings and documents and visualizes the actual decisions taken by the group.

Samenvatting (in Dutch)

"Requirement Elicitation" is één van de eerste fasen in de ontwikkelingscyclus voor alle software. Serious Games zijn hierop geen uitzondering; een grondige probleemanalyse is nodig voor het ontwerpen van een Serious Game, met name vóór het definiëren van het scenario voor het spel. Op dit moment bestaat er geen gestructureerde aanpak en zijn er slechts enkele richtlijnen over hoe deze fase aan te pakken. Het ontwikkelen van games (en dus ook Serious Games) wordt vooral beschouwd als een zeer creatief proces. Hoewel het ontwikkelen van games een aanzienlijke hoeveelheid creativiteit vereist, kan de ontwikkeling van deze games profiteren van een meer gestructureerde en begeleide ontwikkeling, om twee belangrijke redenen: de ontwikkelingskosten te verminderen en project falen te voorkomen.

Het doel van dit proefschrift is om de "Requirements Elicitation" van Serious Games te ondersteunen. Omdat verschillende mensen (met verschillende achtergronden) betrokken zijn bij het ontwikkelingsproces van een Serious Game, streven wij naar een eenvoudige en makkelijk te gebruiken tool. We hebben daarom gekozen voor een visueel applicatie op een (iPad) tablet.

Dit proefschrift begint met een onderzoek naar mogelijke grafische applicaties die momenteel beschikbaar zijn voor een iPad tablet. Het tool moet een gebruiker, die niet volledig vertrouwd is met het probleem domein van games definitie, kunnen begeleiden bij het nemen van de nodige beslissingen. De thesis geeft vervolgens een voorstel voor het vastleggen van het beslissingsmodel met behulp van een feature-gebaseerd model en presenteert een grafische gebruikersinterface om de gebruikers te helpen bij het vastleggen van de vereisten.

In het tweede deel geeft het proefschrift technische details over de implementatie van de "GuideaMaps" prototype applicatie. Deze iOS-app draait op iPads en helpt de gebruiker bij het verzamelen van de benodigde informatie en het besluitvormings-proces.

De grafische applicatie stuurt de verschillende gebruikers die betrokken zijn bij de ontwikkeling van een Serious Game (bv. tegen cyberpesten) en laat hen nadenken over de doelen, de kenmerken en de belangrijkste principes van de nieuwe Serious Game die ze willen ontwikkelen. De tool kan worden gebruikt tijdens vergaderingen en documenteert en visualiseert de feitelijke beslissingen van de groep.

1 Introduction

1.1 Context and Problem Statement[1]

More and more, game technology is used for purposes other than entertainment. This domain is called Serious Games (Susi, Johannesson, & Backlund, 2007). One application area for games currently investigated at the research lab WISE[2] of the Vrije Universiteit Brussel is the domain of health intervention, and more in particular interventions against cyber bullying (Vandebosch & Van Cleemput, 2008; Walrave, Demoulin, Heirman, & Van der Perre, 2009). Cyber bullying (bullying via electronic communication tools) is a relatively recent phenomenon that especially occurs among early adolescents. As cyber bullying may have a serious impact on the mental (and physical) well being of victims it is important to develop effective evidence-based interventions against cyber bullying.

The "Friendly ATTAC"-project (IWT, 2012), a research project carried out by WISE together with the University of Antwerp (MIOS), Ghent University (Research group Physical Activity, Fitness and Health), and HOWEST (ELIT), has the aim to develop digital games to modify behaviour patterns associated with cyber bullying. The target audience consists of early adolescents (10-15 year old). The games will allow youngsters, through the use of the virtual scenarios, to experience different roles (bully, victim, or bystander) in cyber bullying incidents, to react to those experiences, and to get adjusted feedback based on their individual reactions. In this way, it is hoped to make youngsters more aware of the consequences of certain behaviour (as bully or as bystander) and help them to prevent becoming a victim. Bullies could be encouraged to cease bullying by increasing their empathy, victims could be taught adequate coping strategies and bystanders could be sensibilized to intervene.

To realise the objectives of the Friendly ATTC project, an interdisciplinary team is established consisting of social scientists, health psychologists, computer scientists, and game designers. To allow all these people with different background to collaborate in the design of the games, a domain-specific modelling language is currently developed.

Such a domain specific modelling language uses a dedicated vocabulary and provides abstractions that make the specifications of solutions easier and more accessible for non-technical people. In general, domain-specific modelling

[1] http://wise.vub.ac.be/content/ipad-brainstorming-tool-development-serious-games

[2] http://wise.vub.ac.be

languages are graphical (visual) languages because graphical or visual specifications are easier for the communication with non-technical people than textual languages; they are also helpful for conveying complex models and designs as they can help people to grasp large amounts of information more quickly than large listings of text.

The domain specific modelling language will be used to define the scenarios for the games. However, before scenarios can be defined for a game a lot of decisions need to be taken, for example, on which role to focus (victims, bullies, or/and bystanders)? Or the specific age range of the target users? The original target group, 10-15 year old, is still very broad; games appealing for 10-year old children may not be appealing for 15-year old adolescents. Also the platform on which to offer the game (PC, tablet, smartphone, the Web) is important, as well as the availability of the game (only in class room, in closed environments, or publicly available), the genre of the game, the goal of the game, the type of scenario, and so on.

In the context of the Friendly ATTAC project, plenary sessions were held to collect ideas for the games to be developed (i.e. for the requirement elicitation). Different plenary sessions were held. Some sessions were held only with the members of the Friendly ATTAC team consisting of researchers from social science, health psychology, computer science, and teachers/researchers in game development. Other sessions were together with members from the user advisory board of the project, which includes different types of stakeholder: educational/youth stakeholders, e-safety stakeholders, heath promotion stakeholders, and technological stakeholders.

During those sessions, many different ideas and issues were raised. There were discussion about the age range of the target users; gender issues; on which role to focus (victims, bullies, or/and bystanders); the platform on which to offer the game (PC, tablet, smartphone, the Web); the availability of the game (only in class room, in closed environments, or publicly available); the embedding of the game in learning environments and other learning materials; the embedding in social networks; the involvement of teachers, parents, friends during playing and coaching issues; issues about risks (not being inspiring for bullies) and privacy; the duration of the game; the genre of the game; the use of mini games; the combination of the game with real life assignments; the use of a scenario dealing explicitly with cyber bullying versus a scenario that is not directly related to cyber bullying and actually hides the pedagogical message inside entertainment); learning styles of children; type of feedback; motivation for playing the game, and much more.

Although the plenary sessions were quite successful in generating a lot of interactions and issues to consider, it was difficult to come to decisions. An important lesson learned by the research team is that a thorough problem analysis is needed before starting to design a serious game, and in particular before defining the scenario for the game. Creating a serious game is not only about defining an attractive scenario. Before this can be done, one has to decide or clarify a lot of other issues that could (or not) influence the scenario. Although a lot of relevant information can be found in different publications, as far as we known, a concrete list of such issues and alternatives to decide on is not readily available. Such a list could have made the discussions more focused and efficient. There is also no dedicated tool that could help making such decisions.

Therefore, the purpose of this thesis is to develop a tool that allow the different people (and with different background) involved in the development of a serious game (e.g., against cyber bullying) to brood over the goals, characteristics and main principles of a new to develop serious game. The tool should be easy to use and usable in meetings. Therefore, we want to explore the characteristics and capabilities of a tablet (i.e. iPad).

1.2 Research Questions

Based on the problem statement described in the previous section and the objectives of the thesis, we have formulated the following research questions.

1.2.1 Main Question

> *RQ1: How to guide non-technical people (i.e. casual users) in the requirement elicitation process of a serious game using (iPad) tablets?*

When outlining a serious game, a number of decisions need to be made regarding the purpose and characteristics of the game. We refer to this process as the process of *the requirement elicitation process for the serious game*.

1.2.2 Sub-Questions

To answer the above question, a number of sub-questions can be formulated:

> *RQ1.1: What are the available applications on iPad that could support the goal formulated in RQ1 and could they be customized to be usable in the context of serious games?*

Mainly two types of tablets currently exist: Apple's iPad[3] and the Android[4] tablets, each with their own development environment. It is not yet possible to develop a tablet application that can run on both types of tablets; therefore we decided to focus on the iPad. There is no particular reason for the choice of the iPad over an Android tablet.

Apple's iOS App Store and iTunes App Store are the primary software distribution vehicles for finding and installing applications for the iPad. Many applications are available; it may be possible that a tool exist that satisfies our requirements or can be customized to the context of serious games.

> *RQ1.2: What are the different decisions and choices that should be made when defining a serious game?*

In order to be able to support non-technical people in the requirement elicitation process for serious games, we need to known which decisions need to be taken during this process and what alternatives or options are available. Dependencies and relationships between alternatives (both inclusive and exclusive) should also be outlined.

> *RQ1.3: How can we capture and store the different aspects to decide on, the related options and their dependencies in a structured way?*

In computer science, a structured way to capture information is called a model. Here we will use the term *decision model* to refer to the model that captures the different aspects to decide on, the related options and their dependencies.

> *RQ1.4: How can we use or create a tool in a generic way so that new options and alternatives can be added when necessary or a new model can be used (e.g., for a different purpose)?*

New information and insights may become available later on. Therefore, we want to opt for a tool that can be easily extended or modified. It should be able to use supplementary decision models in other contexts, e.g., serious games for math learning or serious games for practicing social skills. This may have an impact on the decisions to take and the alternatives available.

> *RQ1.5: Is a tablet (i.e. iPad) applicable for realizing the above goals?*

The software should be able to display the (possible) complex decision model (resulting from RQ1.3). The user should be able to retrieve the right level of

[3] http://www.apple.com/ipad/

[4] http://www.android.com

information and be able to get feedback on the options in a simple but flexible way. As we have little experience with developing tablet apps and their limitations, it is a question whether we will be able to realize this using the tablet technology.

> *RQ1.6: What is a usable UI on a tablet to visualize the choices and alternatives in the requirement elicitation process and how can it guide and support the user through this process?*

Game definition information can be presented in a textual representation using typical “master-detail” navigation screens. However, a tablet offers new interaction techniques, e.g., gestures techniques to provide an easy and fast navigation. We aim to investigate how we can exploit the possibilities offered by tablets to provide non-technical users an easy to use UI.

1.3 Proposed Solution

To answer RQ1 (How to guide non-technical people (i.e. casual users) in the requirement elicitation process of a serious game using (iPad) tablets?), we propose the use of an "ideation" tool. Ideation (Goldenberg, Mazursky, & Solomon, 1999) is the process of creating new ideas about products, services, advertising, systems, etc.

By using a structured ideation tool, the decisions that need to be made regarding the purpose and characteristics of the game can be captured and documented in a well-defined way. Ideally, the tool guides the users (who doesn't have to be computer experts) through the decision process and constantly provides feedback about the choices and alternative options.

Such a tool can be positioned as a "structured mind-mapping" tool with predefined "choices", in which, during "brainstorming" sessions with the team, the decisions can be visualized and fine-tuned. By providing an easy to use "point, click and drag" user interface on a tablet, any user with limited exposure to computers and software should be able to work with the application and define the requirements of the game.

In a typical brainstorming approach, the focus is on generating as many ideas as possible: "Brainstorming is an idea generation technique that focuses on using quantity to breed quality" (Shih, 2011).

1.4 Approach

To answer the above research questions in a structured way, we have adopted the **Design Science Research Methodology** (DSRM) (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007) that incorporates principles, practices, and procedures required to carry out research. It includes six steps: problem identification and motivation, definition of the objectives for a solution, design and development, demonstration, evaluation, and communication.

1.4.1 Problem Identification and Motivation

The problem identification and motivation have been covered by section 1.1 "Context and Problem Statement" and section 1.2 "Research questions".

1.4.2 Definition of the Objectives for a Solution

In order to define the objectives for a solution, we identified a set of high-level functional and usability requirements that must be fulfilled by potential solutions.

A study will be made to investigate the above questions and provide more insight in the domain of "serious games" ideation to answer RQ1.1 and RQ1.2.

If the outcome of the study concludes that no "guided ideation" tool currently exist for defining "serious games" on iPad devices, a prototype of a new tool will be defined and implemented. The technical specifications will be described and investigated on how to build this new application from scratch that matches the above requirements.

1.4.3 Design and Development of a Solution

The proposed tool will be implemented as a native iOS application. The tool will aim to satisfy the objectives of the solution as defined in research questions and the high-level requirements.

A number of issues need to be resolved:

- A mechanism for defining the decision model needs to be investigated and implemented to answer RQ1.3 and RQ1.4.
- A structure for storing the user's choices from a given decision model needs to be defined and implemented.
- A graphical representation of the decision model needs to be prototyped and realized. Based on a number of (paper and coding) prototyping iterations that are matched against the usability requirements and user feedback, a final User Interface will be selected and implemented. This will provide an answer RQ1.6.
- The application needs to be implemented using the native tools and frameworks available on iOS devices to answer RQ1.5

1.4.4 Demonstration

In order to demonstrate the usefulness of the proposed solution, a decision model will be defined that matches the requirement gathering choices when defining a Serious Game. This decision model will be converted into a structure that is readable by the implemented iPad application and visualized using the chosen User Interface.

The user will be able to select the different choices as defined in the decision model and immediately get feedback on his actions. The user can manipulate the decision model for the Serious Game as defined in the predefined usability requirements.

1.4.5 Evaluation

The evaluation of the solution will be done at different occasions:

- During the ACM SIGCHI Conference on Human Factors in Computing Systems at Paris, the tool will be presented during a workshop "Let's talk about Failures: Why was the Game for Children not a Success?"[5] (De Troyer, Janssens, & Vandebosch, 2012). This conference is the premier international conference on human-computer interaction.
- A presentation will be organized for the Friendly ATTAC project to gather feedback and input on how to improve the offered solution in future releases.
- A number of decision models have been prepared and successfully visualized in the tool.

1.4.6 Communication

The tool has been presented to the Friendly ATTAC group and to a workshop "Let's talk about Failures: Why was the Game for Children not a Success?" of the SIGCHI conference. We plan to present the tool to (serious) game developers to investigate their interest in the tool. We will also investigate submitting the application to Apple's App Store so it can be installed on iOS devices around the world.

1.5 Thesis Structure

This section gives an overview of the structure of this thesis document.

In the chapter "**Requirements**", high-level requirements are defined (both functional and technical) that are essential for the tool.

The chapter "**Related Tools and Work**" documents the research done to investigate RQ1.1 ("What are the available applications on iPad that could support the goal formulated in RQ1 and could they be customized to be usable in the context of serious games?"). It provides an overview of candidate apps and stipulates more details regarding their advantages and limitations. In addition, it also investigates other available and related research work that could be useful for identifying a solution for the formulated research questions.

The chapter "**Feature-based Modelling to Represent the Decision Model**" provides answers to RQ1.3 to RQ1.5 regarding the decision model. It also proposes a sample Decision Model for a Serious Game as requested in RQ1.2.

[5] http://workshops.icts.sbg.ac.at/CHIGames2013/

The chapter “**GuideaMaps: The iPad Application**” explains the mapping between the feature-based model and the “look and feel” of the User Interface of the iOS-based application and its interaction options. It provides an answer to RQ1.4 to RQ1.6

The “**Implementation Details**” chapter contains in-depth development and coding information about the implemented iPad application.

The “**Validation & Future Work**” chapter documents the validation of the project and research effort and provides a list of possible further work.

The final chapter concludes with the findings and results from this thesis.

2 Requirements

Based on the experience gained in the Friendly-ATTAC project, the following high-level functional requirements (section 2.1) and usability requirements (section 2.2) have been formulated for the tool.

2.1 Functional Requirements

The main functional requirements are summarized in the following list:

1. The user should be able to take the required decisions regarding the purpose and characteristics of a serious game.
2. The tool should guide the user, using a predefined set of choices, through this problem analysis process.
3. The app should visualize the available game definition choices and capture decisions by means of an easy to use graphical user interface.
4. The tool should provide explanations for the different choices and options. This is necessary as not all people involved will be familiar with the definition process of serious games.
5. The tool should distinguish between choices that are required to consider and choices that are optional (because they may not be applicable for the case at hand).
6. The tool should provide the possible choices and alternatives when decisions need to be made for an issue. Again, this is necessary, as not all people involved in the process will be familiar with games or serious games.
7. The tool should allow motivating choices and issues considered. This will allow documenting the process.
8. The tool should capture the motivations for the issues considered in an easy to understand graphical user interface.
9. The tool should indicate the impact of choices. The choice of an option or alternative may have an impact on the options and alternatives available for other options. For instance, the choice for a certain pedagogical approach may limit the choice for the game genre. The user should be able to change decisions already made and view the alternative choices again.
10. The tool should be generic, meaning that it can be used for different types of games, with different choices and options, i.e. it can be used with different decision models.

Optional Requirements

A number of additional requirements can be identified that are out-of-scope for the initially version of the iPad application. For the sake of completeness we also provide them:

12. The user should be able to store multiple game requirements in different "documents" that are stored locally on the iPad device or remotely in Apple's iCloud.
13. The user should be able to share game requirements documents with other users using Dropbox integration.

2.2 Usability Requirements

The main usability requirements are summarized in the following list:

1. The tool should be usable in meetings and by different types of people (which are in general not schooled in computer science, i.e. so-called casual users).
2. The user should be able to start using the app immediate after installation; a simple metaphor will allow the user to explore the choices that are presented in a tree-based structure.
3. The user should be able to easily switch between different decision models and requirements documents working on.
4. The application will store the state of the requirement elicitation process; when the user restarts the app, it will display the same information as when it was stopped.
5. The tool should allow exporting the results in a textual and readable form.
6. The default decision model that describes the different options and choices is included in the application; additional decision models can be installed if required.

3 Related Tools and Work

This chapter provides an answer to the research question RQ1.1: “What are the available applications on iPad that could support the goal formulated in RQ1 and could they be customized to be usable in the context of serious games?”.

An overview of such related tools is provide in section 3.1. We will also discuss related work from the literature in section 3.2.

3.1 Ideation Apps on iPad

An analysis was performed to get an overview of the “ideation” apps that are currently available for iPad. The purpose of the study was twofold: which existing apps match the defined requirements and, in case none of them do, what concepts and ideas can be “borrowed” from these tools to implement a custom app for “guided ideation”.

The tools were selected by querying Apple’s App Store and looking for apps with keywords “ideation”, “brainstorm” and “mind map”. A number of applications were identified that potentially matched some of the predefined requirements.

1) **iBrainstorm**[6] [Free app]
iBrainstorm (figure-3.1) is a multi-device collaboration tool; up to 4 people get the ability to create ideas on their iOS devices and “flick” them over to each other. The app is a generic sketching tool and uses an integrated notes concept. The user can add/edit “post-it”-like notes and move them around on the screen. No ideation guidance functionality is provided.

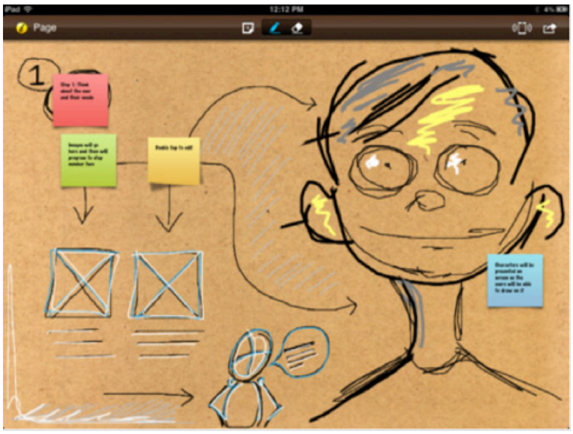


Figure-3.1 iBrainstorm

[6] http://www.universalmind.com

2) **iMindMap HD**[7] [Free App]
   iMindMap HD (figure-3.2) provides a workspace for brainstorming ideas and note taking based on a set of predefined "templates" that can be completed and extended. It offers templates for meeting minutes, weekly planner, note taking etc. with icons and placeholder for entering text.

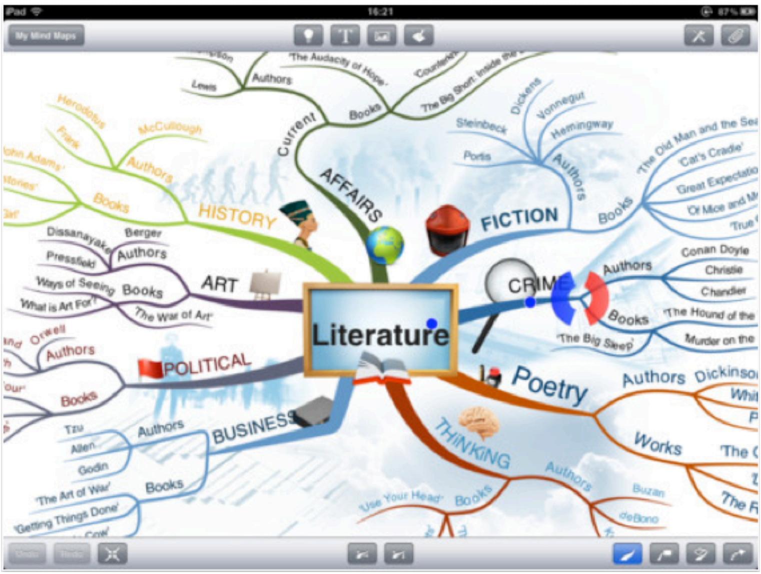


Figure-3.2 iMindMap HD

Nodes can be collapsed/expanded while the full version (available via in-app subscription) adds floating text and ideas, export to PNG/PDF and tidy up functionality.
<u>USEFUL FEATURE(S)</u>: The idea of **templates is a key concept** for the building Guided Ideation tools that can drive the user's choices.

3) **Popplet (lite)**[8] [Free app, full version 4,49€]
   Popplet (figure-3.3) provides a simple interface to capture ideas and sort them visually in real-time. It is a combination of a free-form brainstorming and mind-mapping tool that can also be used to draw diagrams, lists and process charts. It provides a boundless board, pan and zoom support and PDF/JPEG export.
   A full version is available in which the user can create an unlimited number of "popplets" (diagrams).

[7] http://www.thinkbuzan.com/be

[8] http://popplet.com

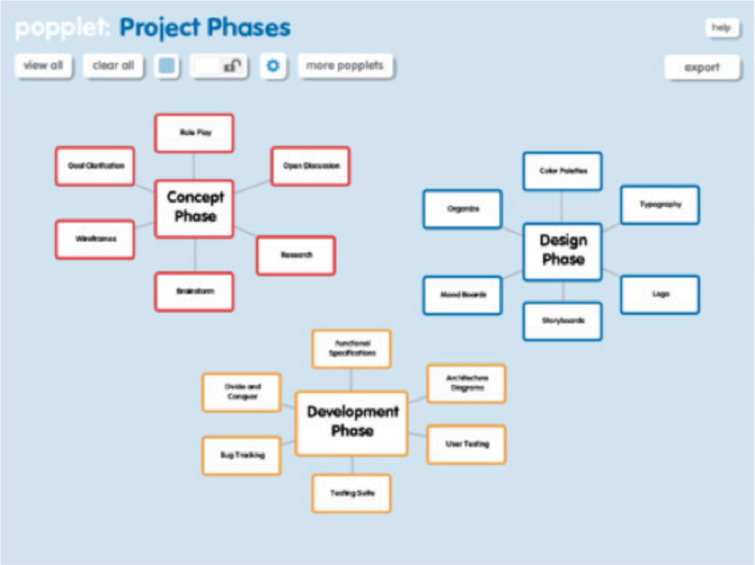


Figure-3.3 Popplet

The **UI for adding and modifying parts is very slick**.

<u>USEFUL FEATURE(S)</u>: Click within the element's rectangle to "select" it; a number of small buttons are displayed to visualize the actions that can be performed on the element: changing the background colour, the text, pen drawing and image inclusion.

4) **SimpleMind+**[9] [Free App, full version 5,99€]
SimpleMind+ (figure-3.4) is a typical mind-mapping tool with easy to use drag, arrange and edit "thought topics".
Mind maps can be exchanged with the desktop version of the tool on Mac OSX and Windows.
A "full version" includes cross-links, collapse/expand, predefined icons, search, etc. and can be obtained via In-App purchases.

[9] http://www.simpleapps.eu/simplemind

The tool provides an intuitive interface with selectable topics.

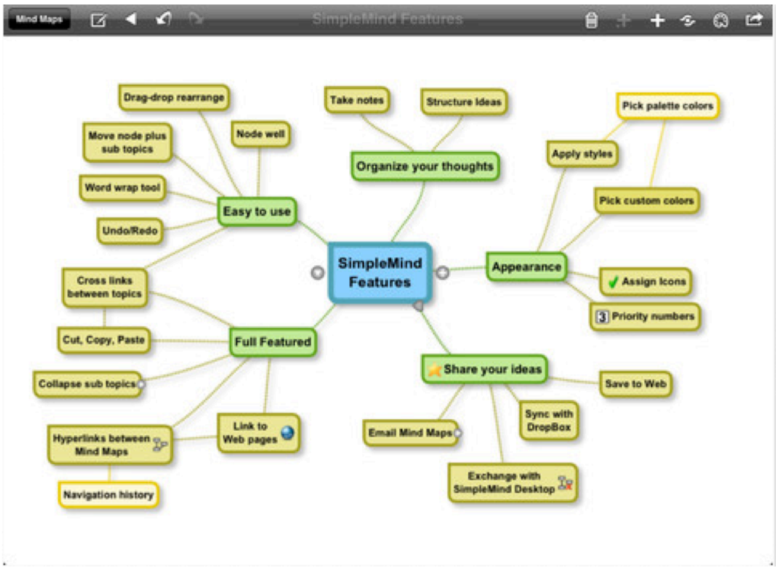
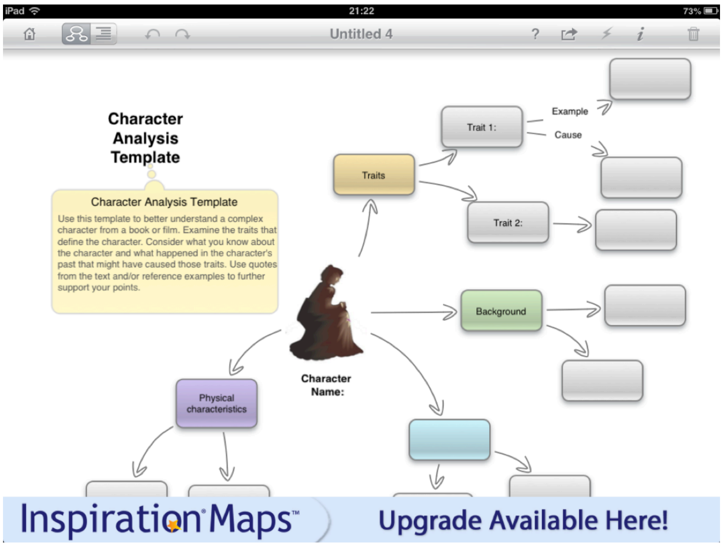


Figure-3.4 SimpleMind+

The concept of **selecting an individual topic and add subtopics** in an intuitive way is providing a good user experience but is less relevant in guided ideation where the choices are predefined in the decision model.

5) **Inspiration Maps (Lite)** [10] [Free App, full version 8,99€]
Inspiration Maps (figures-3.5/3.6) is a template-based brainstorming, mind-mapping and note-taking application. It includes a set of templates for biographies, character and historical event analysis, essay outlines, etc.



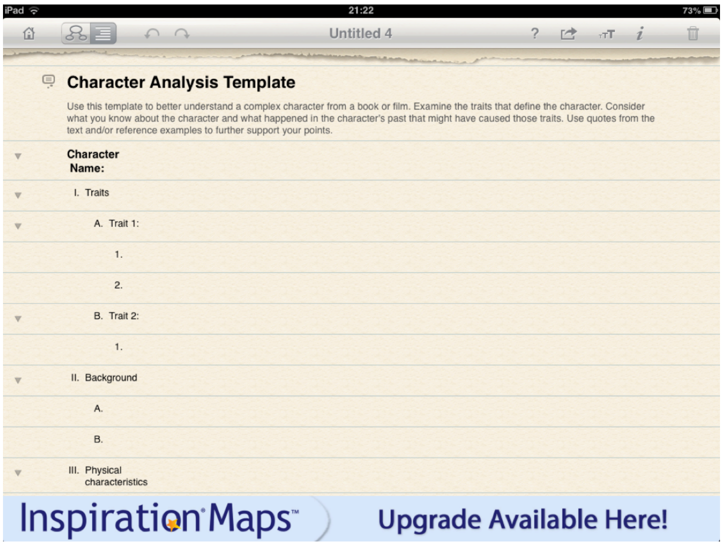
Character Analysis Template

Use this template to better understand a complex character from a book or film. Examine the traits that define the character. Consider what you know about the character and what happened in the character's past that might have caused those traits. Use quotes from the text and/or reference examples to further support your points.

Figure-3.5 Inspiration Maps

[10] www.inspiration.com

One single tap transforms the visual representation into an outline view that can be used to export and include in other applications.



Character Analysis Template

Use this template to better understand a complex character from a book or film. Examine the traits that define the character. Consider what you know about the character and what happened in the character's past that might have caused those traits. Use quotes from the text and/or reference examples to further support your points.

Character Name:

I. Traits

A. Trait 1:

1.

2.

B. Trait 2:

1.

II. Background

A.

B.

III. Physical characteristics

Figure-3.6 Inspiration Maps in Outline mode

The user can personalize the images, colours and shapes and attach notes to any item to start writing. It integrates with their Inspiration 9.1 desktop application.

USEFUL FEATURE(S): Switching between the graphic form and outline can be very useful when exchanging information between applications.

6) **DropMind Lite** [11] [Free App, full version 10,99€]

DropMind (figure-3.7) is a mind mapping application to visually capture, organize and store ideas.

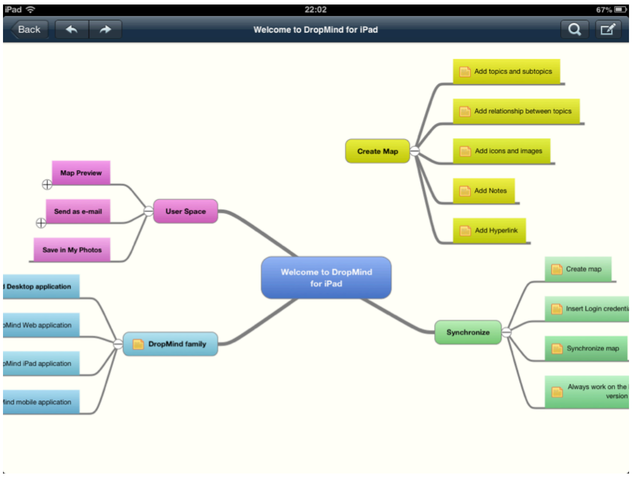
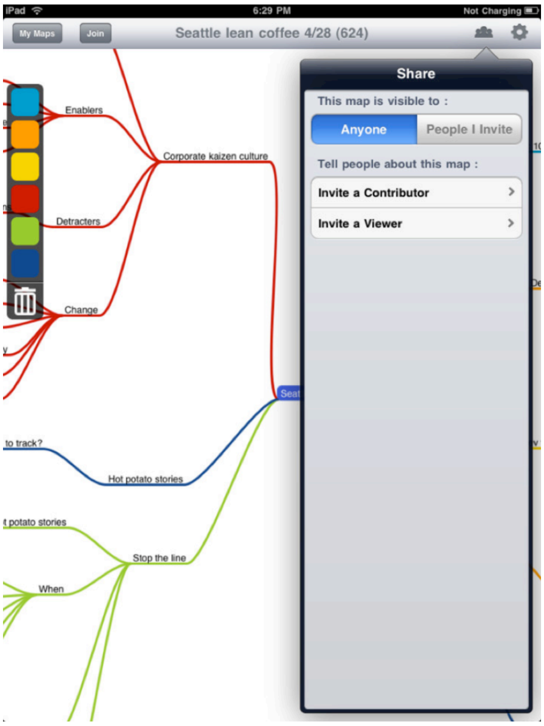


Figure-3.7 DropMind Lite

[11] www.dropmind.com

The product integrates with Dropbox and allows map sharing using the following formats: dmmx, image, pdf and outline.
The company provides mind-mapping tools for desktop and web.

7) **Maptini** [12] [Full version 4,49€]
Maptini (figure-3.8) lets the user build mind maps collaboratively, syncing data in real time with friends and colleagues. It includes capabilities to share maps with remote viewers and invite contributors in real time.



This map is visible to :

Anyone | People I Invite

Tell people about this map :

Invite a Contributor >

Invite a Viewer >

Figure-3.8 – Maptini

USEFUL FEATURE(S): The UI is fairly simplistic but the real time contribution and sharing might be a very handy feature.

[12] www.maptini.com

8) **Mindo** [13] [Full version 6,99€]

   Mindo (figure-3.9) is a mind-mapping tool for iPad that can be used for brainstorming and note taking.

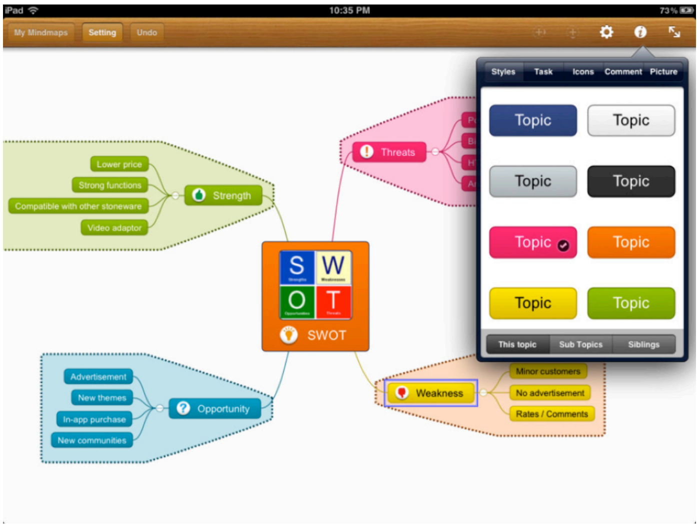


Figure-3.9 Mindo

   <u>USEFUL FEATURE(S)</u>: The visualization of the grouping of related topics and subtopics might be useful to improve the overall readability of the decision model.

9) **iThoughtsHD** [14] [Full version 8,99€]

   iThoughtsHD is a mind mapping tool for iPad that import/export to most popular applications. It exports directly to PowerPoint, Keynote, Word and Pages. The app has integration with Dropbox and sharing capabilities with Twitter and Facebook. It includes a large set of icons, images and background patterns.

Other “ideation”, “brainstorm”, “mind-mapping” Apps

A number of other applications that were included in the App Store query result list were examined. The applications were very similar to the above ones and didn’t reveal any new useful features. These apps were:

- MindGenius for iPad [15] [Free App]
- MindMeister[16] [Monthly subscription]

[13] www.laterhorse.com

[14] www.ithoughs.co.uk

[15] www.mindgenius.com

- Total Recall[17] [Free App]

Outcome:

Apple's App Store provides a wide range of applications that are labelled with "Brainstorming" or "Mind Mapping". However, no "Guided Ideation" applications currently exist that match the above requirements; a lot of applications are glorified mind-mapping tools that include some level of template support. The templates can be used as a short cut for the mind-mapping process but not as a mechanism to drive the user and guide him with the possible choices and implications of his requirement elicitation task.

Therefore, a custom-made iPad app will be developed, based on the specified requirements. Some useful features and ideas can be "borrowed" from the examined apps:

- The use of templates as a kick-starter (as example of a decision model)
- The inclusion of (small) buttons directly in the UI object for triggering actions on that object (compared to a separate menu or dialog)
- The outline view for sharing a model with other applications over email

3.2 Related Work

In this section we consider related work from the research literature. We searched for work considering the support of the development of (serious) games in the early phases of the development and targeted towards casual users, but little was found in this context.

In (Agustin et al., 2007) the authors wanted to deal with the observation that "The exploration of new games in this day and age is thus hampered by the cultural forces of risk aversion, the merit-ocracy reward system, and technology focused on production rather than exploration". They propose game sketching as a way to explore new ideas in a fun, cheap, and risk-free manner. A similar sketching approach is presented in (Smith & Graham, 2010). Although these approaches also focus on the early development phase, the focus is on the ideation of the gameplay rather than on requirements elicitation.

In (Kultima, Niemelä, Paavilainen, & Saarenpää, 2008), idea generation games to be used by game designers are introduced. The purpose was to enhance the creative process by immersing people into a playful activity. They aimed to

[16] www.mindmeister.com

[17] www.appannie.com

support the creation of casual mobile multiplayer games. They proposed three different card games. A similar idea was proposed in (Duin, Hauge, & Thoben, 2009) where the refQuest game is proposed to structure the ideation process in the very beginning of an innovation process. The game is based on the Synectics method (Gordon, 1961), and the process used is very similar to brainstorming but presented as a game. Although using game principles for supporting the ideation process is interesting, at this moment we didn't opt for this direction as we aimed for a simple solution usable by casual users. It is not sure that all kinds of casual users would be prepared to play games for the purpose of requirements elicitation.

In (Vaajakallio, Lee, & Mattelmäki, 2009), two experiments are described to explore the applications of co-design methods with children. For the first experiment, they were using Make Tools[18] (Sanders, 2006) and for the second the Design Games (Brandt & Messeter, 2004) approach. Making Tools's aim is design innovation by exploring collective creativity and is too open ended for our purpose. The aim of Design Games is to help facilitate a user-centered design process for cross-disciplinary design groups early in the design process. This approach also uses the concept of game and play to structure the design activities.

EMERGO (Nadolski et al., 2008) is a methodology and toolkit for developing serious games. For the analysis phase, EMERGO provides a list of questions to be answer to obtain a global description of the intended case. The questions are grouped into subjects. Although the list of questions provided are relevant for our purpose (i.e. decision model), it is not clear if the toolkit is supporting this phase is some way.

Carro et al. (Carro, Breda, Castillo, & Bajuelos, 2006) also provide a methodology for supporting the design of adaptive educational game environments. They established a set of steps to be followed, such as identifying the types of users, specifying the game goals, etc. Parts of these steps can be considered as the requirement elicitation that we aim for. They also developed a model for describing this information. Also this information is useful for our purpose (i.e. decision model), but again it is not clear if and how a tool supports this.

In "A Channeled Ideation Approach" (Goldenberg et al., 1999), Goldenberg proposes a template-driven approach when defining new products. They divide products into different components and their attributes. This approach can be reused when defining a decision model as "products". These templates define the possible combinations and their constraints.

[18] www.maketools.com

Furtado (Furtado, 2006) considers software factories (i.e. software product lines) and domain specific languages for the development of games. Our solution also uses the concept of software product lines, but Furtado uses a software product line for the actual development of a particular type of game, while our approach uses it to do the requirement collection of a (serious) game.

4 Feature-based Modelling to Represent the Decision Model

In order to answer RQ1.3 (“How can we capture and store the different aspects to decide on, the related options and their dependencies in a structured way?”) a structured modelling methodology is needed to outline (i.e. model) the different characteristics of the Serious Game and define the different relationships between them. The resulting model will be used to guide the user in making decision and selecting options for delimiting his game.

In this work, we represent the decision model as an XML-based feature model. To motivate this decision, we first describe the concept of feature model.

Kang introduced the concept of Feature Oriented Domain Analysis (FODA) (Kang, Cohen, Hess, Novak, & Peterson, 1990) within software development. FODA includes the concept of a *feature model*, which is a compact representation of all the products of the Software Product Line (SPL) in terms of "features". Feature models are visually represented by means of feature diagrams. Feature models are widely used during the whole product line development process and are commonly used as input to produce other assets such as documents, architecture definition, or pieces of code.

A "feature" is defined as a "prominent or distinctive user-visible aspect, quality, or characteristic of a software system or system". The focus of SPL development is on the systematic and efficient creation of similar programs. FODA is an analysis devoted to identification of features in a domain to be covered by a particular SPL.

Figure 4.1 shows an example (Cechticky, Pasetti, Rohlik, & Schaufelberger, 2004) of a feature model diagram,. This model defines features and their dependencies, typically in the form of a feature diagram and cross-tree constraints.

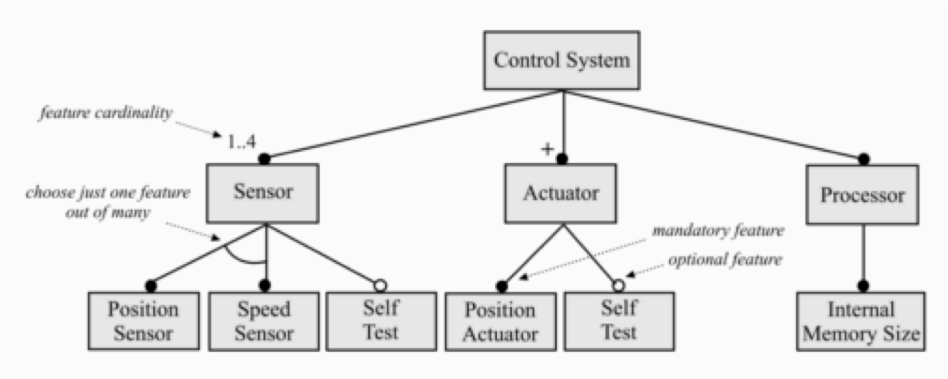


Figure-4.1 Visual Representation of a Feature Model

A feature diagram is a visual representation of a feature model, which is basically an and-or tree. It can include information about cardinalities, feature cloning, feature attributes, etc.

A *configuration* is a set of features that describes a member of an SPL: the member contains a feature if and only if the feature is in its configuration. A feature model permits a configuration if and only if it does not violate constraints imposed by the model.

The feature model concept can be applied to the Decision Model that is required for our solution. The feature model can define the different elements and choices that the user can make to define the requirements of his Serious Game. The decisions and choices taken can be considered as a configuration. We will call such a configuration the *Application Model*.

Cechticky proposes an XML-based approach to expressing the feature models (Cechticky et al., 2004). XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C.

The design goals of XML emphasize simplicity, generality, and usability over the Internet. It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures.

In the paper "Feature Assembly Modelling, A New Technique for Modelling Variable Software" (Zaid, Kleinermann & De Troyer, 2005), the authors introduce a simplified feature modelling technique containing two concepts that fit the needs of the guided ideation tool.

The "Basic Modelling Primitives" of Feature Assembly initiate the "featured type" to express how a feature contributes to variability; a feature can be tagged as "**abstract**" or "**concrete**". An abstract feature can be considered as a placeholder for different options (called option features). It allows expressing a kind of specification relationship. The abstract feature represents a variation point; the concrete features can be used as "option" features of the abstract one. The cardinality constraint can specify the minimum and maximum number of "choices" that can be made. Furthermore, the technique allows expressing composition relations, i.e. a feature can be composed of other features, mandatory as well as optional features. These modelling concepts allow us to define our Decision Model for Serious Games as a feature model using composition and generalization/specification relationships.

The "Feature Dependencies" available in Feature Assembly allow specifying **how features affect other features**; they define which features should exclude others or which features require (depends on) others. This concept allows us to define the proper relationships between the different feature (option)s.

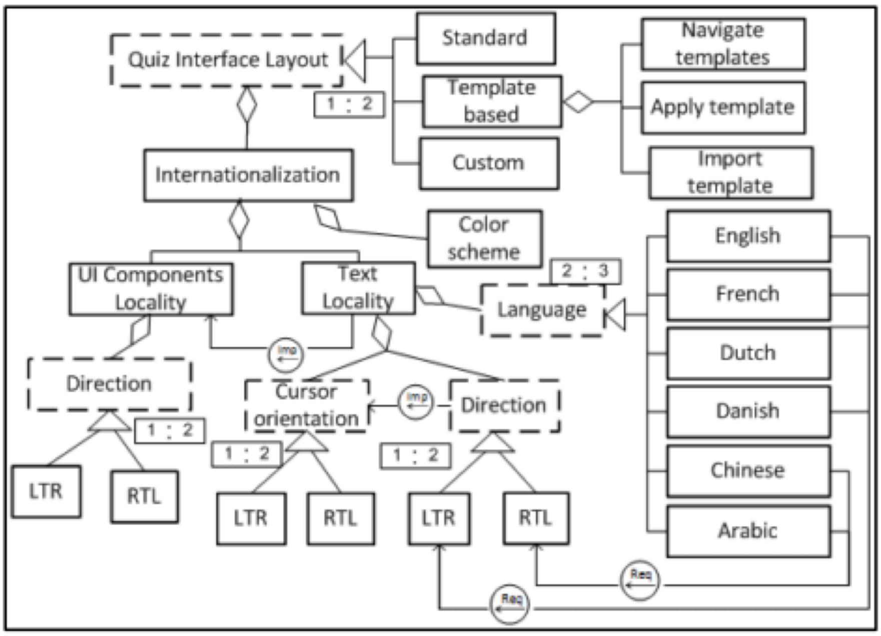


Figure-4.2 Quiz example in Feature Assembling Modelling

In the paper "XML-based feature models" (Cechticky et al., 2004), the authors introduce the concept of a **meta-level representation** of features using an XML Schema Definition (XSD) notation. Their paper makes a distinction between the "**family**" and the "**application**" model. The first one represent all features and their possible relationships, the second a concrete configuration.

This approach is applicable for the ideation tool: the "family" contains all the features and possible relationships (= the Decision Model); the "application" represents the concrete choices for a Serious Game".

By combining the Family/Application concepts and the Feature Assembly Modelling concepts, we can define an easy to read, to maintain and to extend model using standard XML technologies. We define a XSD meta-level representation for the Decision Model that is used in the actual Decision Model, expressed in XML.

From this model, an application perspective is assembled, based on the actual choices and input from the user, i.e. the Application Model.

As the feature model will be represented using XML, standard and mature XML technology (parsers, frameworks) can be used where needed. The XML structure of both the Decision and Application Model are fairly simple as the definition of the different game components and their relationships are straightforward.

4.1 XML Representations of the Models

As explained, XML schemes are specified to represent the structure of the Decision and Application Models. The XML-based models will use these schemes as they describe the allowed document content and can be used to validate the correctness of the data.

A standard XML editor (e.g. Altova XML Spy) can assist in the document creation and validate the XML for syntactical errors. The Decision Model will contain all the options and choices the user can make. Note that these XML-documents will, in general, not be created by the user of the requirement elicitation tool (the app). The Decision Model is created by an person experience in Serious Game development, and an XML-schooled person can be ask to translate the Decision Model into the required XML format.

The Application Model that is also structured using an XML scheme, captures the selections and choices made by the user, i.e. the configuration created. This XML-based model will also keep track of the x/y position of the different elements and the comments that were provided by the user. This XML is called the Application XML and is constructed by the app itself.

4.1.1 XML Schema for the Decision Model

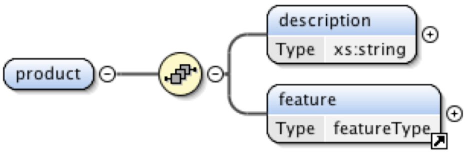
The XML Schema for the Decision Model (see snippet-4.1) defines all the possible XML tags and attributes that can be used for specifying the Document Model content:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="product">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="model" type="xs:string" minOccurs="0" maxOccurs="1" />
        <xs:element name="author" type="xs:string" minOccurs="0" maxOccurs="1" />
        <xs:element name="comment" type="xs:string" minOccurs="0" maxOccurs="1" />
        <xs:element ref="feature" minOccurs="1" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="feature" type="featureType" />

  <xs:complexType name="featureType" mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="cardinality" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute type="xs:string" name="min" use="required"/>
                <xs:attribute type="xs:string" name="max" use="required"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element ref="feature" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element        type="xs:string"        name="requires"        minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element        type="xs:string"        name="excludes"        minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="id" use="required"/>
      <xs:attribute type="xs:string" name="abstract" use="optional"/>
      <xs:attribute type="xs:string" name="extends" use="optional"/>
      <xs:attribute type="xs:string" name="optional" use="optional"/>
    </xs:complexType>
</xs:schema>
```

Snippet-4.1 XML Schema for the Decision Model

The above XSD (XML Schema Definition) defines the different feature characteristics and their inter-relationships. The top element of the XSD is a <product> tag (represented in figure-4.3) that contains a description and a root feature.



Figure–4.3 <product> tag

Figure-4.4 represents the “featureType” complex XML type that contains a set of attributes and XML tags to specify the details of a specific feature and its characteristics.

Every feature has a unique identifier and an indication of whether the feature is abstract or extending an abstract one. The name and description is used as descriptive information about the feature.

A feature can have sub-features (decomposition of the feature) and optionally include a list of required/excluded features.

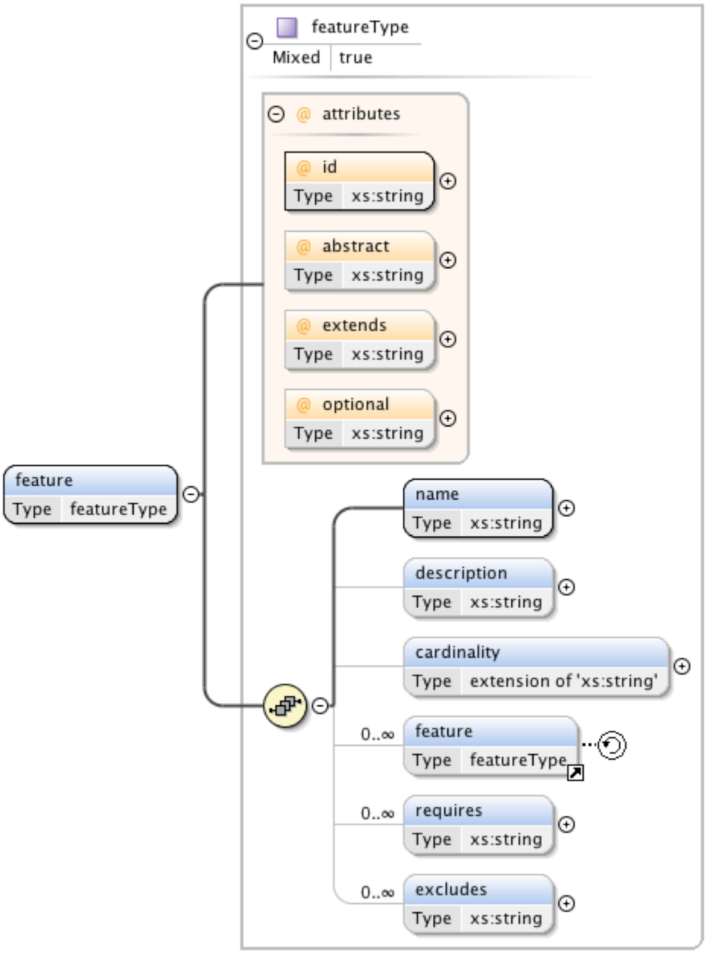


Figure-4.4 featureType complex XML type

The cardinality of the feature can be specified with a minimum and maximum value attribute.

4.1.2 XML-based Decision Model Definition

An XML-based Decision Model is using the featureModel.xsd to describe the complete model. It contains descriptions for all the different features, their attributes and sub-features included in the Decision Model. Abstract features can be specified and a list of concrete choices from which the user can select is included.

Snippet-4.2 represents an example Decision Model.

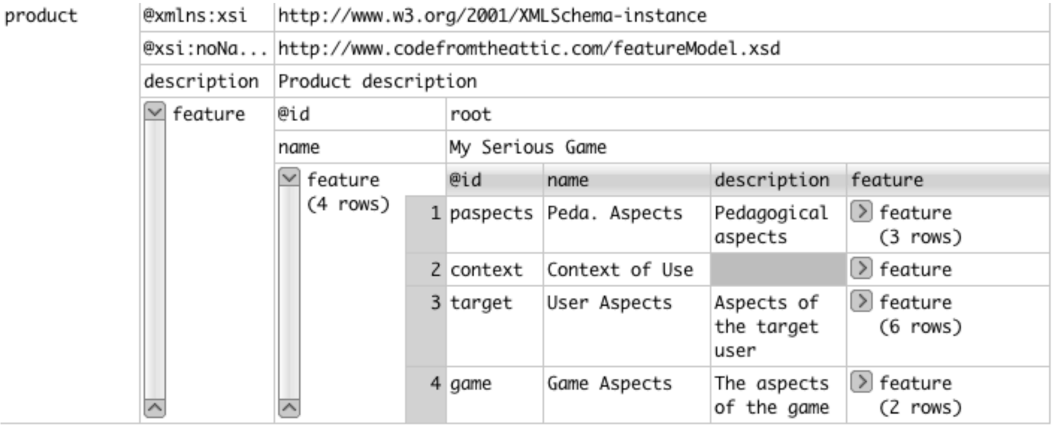
```
<?xml version="1.0" encoding="UTF-8"?>
<product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.codefromtheattic.com/featureModel.xsd">
    <description>Product description</description>
    <feature id="root">
        <name>My Serious Game</name>
        <feature id="paspects">
            <name>Peda. Aspects</name>
```

```
<description>Pedagogical aspects</description>
<feature id="goal" abstract="TRUE">
    <name>Pedagogical Goal</name>
    <description>Select the type of the pedagogical goal</description>
    <feature id="goal1" extends="TRUE">
        <name>Social Problem Solving</name>
        <requires>approach1</requires>
        <requires>context1</requires>
        <excludes>approach2</excludes>
        <excludes>approach3</excludes>
    </feature>
    <feature id="goal2" extends="TRUE">
        <name>Cognitive Problem Solving</name>
        <description>Solve the problem !</description>
        <requires>device2</requires>
        <excludes>context2</excludes>
    </feature>
    <feature id="goal3" extends="TRUE">
        <name>Attitude Change</name>
        <description>Change your attitude</description>
    </feature>
    <feature id="goal4" extends="TRUE">
        <name>Awareness</name>
        <requires>approach4</requires>
    </feature>
    <feature id="goal5" extends="TRUE">
        <name>Practicing Skills</name>
        <excludes>approach2</excludes>
    </feature>
    <feature id="goal6" extends="TRUE">
        <name>Knowledge Gain</name>
        <excludes>approach2</excludes>
        <excludes>approach3</excludes>
        <excludes>context2</excludes>
    </feature>
</feature>
<feature id="approach" abstract="TRUE">
    <name>Peda. Approach</name>
    <description>Select pedagogical approach</description>
    <feature id="approach1" extends="TRUE">
        <name>Practice and feedback</name>
        <requires>goal1</requires>
        <excludes>context2</excludes>
        <excludes>context3</excludes>
    </feature>
    …
```

Snippet-4.2 Example Decision Model

Notice that every feature has a unique **@id** attribute; this is used in the Application Model's XML to point back to an element in the Decision Model.

The above example can be simplified by the following representation in figure-4.5



product

@xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
@xsi:noNa...	http://www.codefromtheattic.com/featureModel.xsd
description	Product description
feature	@id: root name: My Serious Game feature (4 rows)

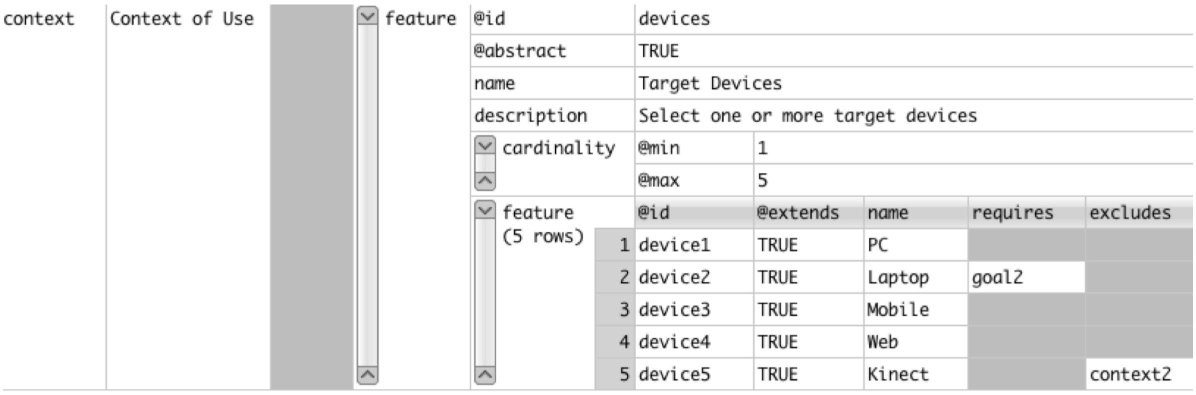
feature (4 rows)

	@id	name	description	feature
1	paspects	Peda. Aspects	Pedagogical aspects	feature (3 rows)
2	context	Context of Use		feature
3	target	User Aspects	Aspects of the target user	feature (6 rows)
4	game	Game Aspects	The aspects of the game	feature (2 rows)

Figure-4.5 Example 1 of a Decision Model

The root feature with **@id=root** is composed of 4 sub-features. Each of them contains one or more sub-features. The **@description** attribute provides a descriptive name for the every single feature.

The feature with **@id=context** is defined as **@abstract=TRUE**; it contains a number of concrete sub-features from which the user should select **@min=1** and **@max=5**.



context | Context of Use

feature

@id	devices
@abstract	TRUE
name	Target Devices
description	Select one or more target devices
cardinality	@min 1 @max 5

feature (5 rows)

	@id	@extends	name	requires	excludes
1	device1	TRUE	PC		
2	device2	TRUE	Laptop	goal2	
3	device3	TRUE	Mobile		
4	device4	TRUE	Web		
5	device5	TRUE	Kinect		context2

Figure-4.6 Example 2 of a Decision Model

In figure-4.6, we visualize that for each feature, one or more constraints can be defined; device2 **requires** goal2, device5 **excludes** context2.

4.1.3 XML Schema for the Application Model

The XML Schema for the Application Model (see snippet-4.3) defines all the possible XML tags and attributes that can be used for specifying the Application Model content.

```
<!--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>
```

Snippet-4.3 XML Schema for the Application Model

The guideaType complex type, as represented in figure-4.7, includes x/y coordinates, colour info and expand status. It also incorporates a comment field and a list of optional guideaTypes.

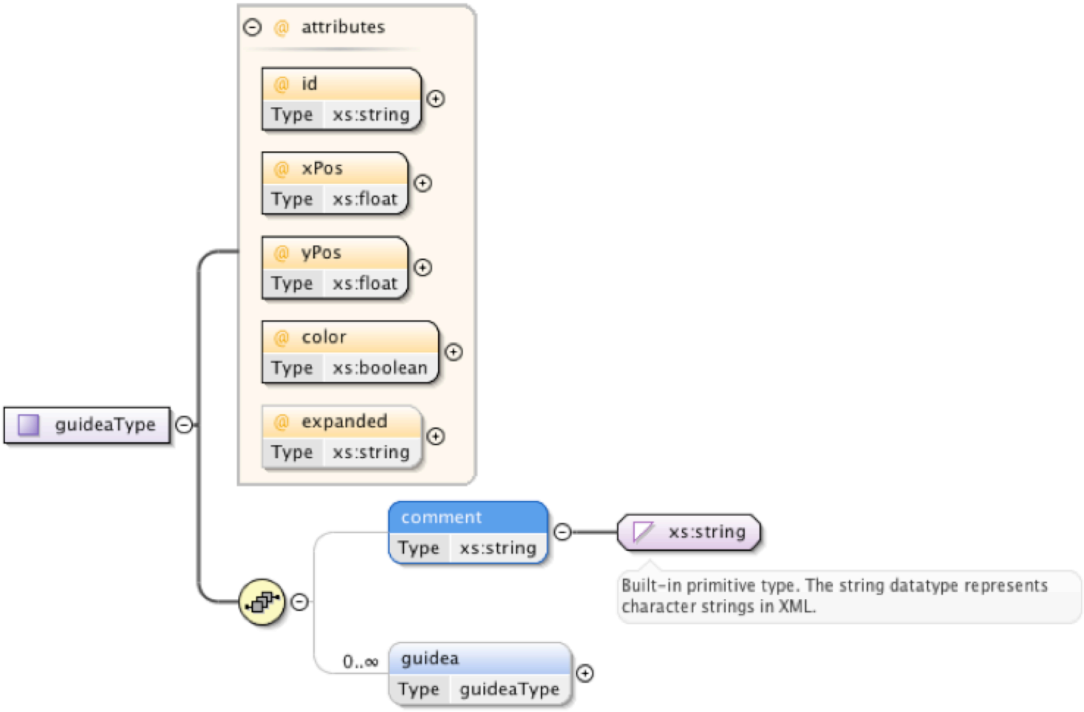


Figure-4.7 guideaType complex XML type

4.1.4 Example Application XML

The Application XML keeps track of the choices that have been made, the x/y position of the different elements and the comments that were provided by the user.

```
<?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>Comment for this Serious Game!</comment>
        <guidea    id="paspects"    xPos="273.000000"    yPos="253.000000"    color="0"
expanded="TRUE">
              <comment>The aspects are defined here.</comment>
              <guidea id="goal1" xPos="357.000000" yPos="23.000000" color="0">
                     <comment>SPV is the recommended goal.</comment>
              </guidea>
              <guidea id="approach1" xPos="87.794922" yPos="135.000000" color="0">
              </guidea>
              <guidea id="context1" xPos="501.205078" yPos="218.000031" color="0">
                     <comment>Home is the best place to learn</comment>
              </guidea>
        </guidea>
        <guidea id="context" xPos="267.000000" yPos="613.000000" color="0">
              <guidea id="device4" xPos="81.000000" yPos="641.000000" color="0">
              </guidea>
              <guidea id="device1" xPos="18.000000" yPos="545.000000" color="0">
              </guidea>
              <guidea id="device5" xPos="58.000000" yPos="609.000000" color="0">
              </guidea>
              <guidea id="device2" xPos="35.000000" yPos="575.000000" color="0">
              </guidea>
        </guidea>
        <guidea id="target" xPos="447.000000" yPos="620.000000" color="0">
              <guidea id="target.age" xPos="298.000000" yPos="575.000000" color="0">
              </guidea>
              <guidea id="target.gender" xPos="124.794922" yPos="675.000000" color="0">
              </guidea>
              <guidea    id="target.characts"    xPos="124.794922"    yPos="875.000000"
color="0">
              </guidea>
              <guidea   id="target.competences"   xPos="298.000000"   yPos="975.000000"
color="0">
              </guidea>
              <guidea   id="target.playertype"   xPos="471.205078"   yPos="875.000000"
color="0">
              </guidea>
              <guidea   id="target.motivation"   xPos="471.205078"   yPos="675.000000"
color="0">
                     <guidea id="social" xPos="356.889709" yPos="741.000000" color="0">
                     </guidea>
                     <guidea    id="immersion"    xPos="585.520447"    yPos="741.000000"
color="0">
                     </guidea>
              </guidea>
        </guidea>
        <guidea id="game" xPos="505.000000" yPos="474.999939" color="0">
              <guidea id="game.goal" xPos="596.000000" yPos="334.999939" color="0">
              </guidea>
              <guidea id="game.genre" xPos="603.000000" yPos="619.999939" color="0">
              </guidea>
        </guidea>
    </guidea>
</guideaModel>
```

Snippet-4.4 XML sample of the Application Model

As shown in snippet-4.4, every id attribute of the **<guidea>** tag refers to the unique **@id** attribute in the Decision Model.

The Application XML includes a unique reference to its Decision Model XML; the **@metaModel** attribute in the **<guideaModel>** tag. Every Decision Model XML should have a unique name so it can be linked to it corresponding Application XML files; it is recommended to use “inverse domain names” to avoid clashes between Decision Models (e.g. com.codefromtheattic.seriousgame).

4.2 Serious Game Decision Model

Based on the input and feedback from the “Friendly ATTAC”-project, a Decision Model for Serious Games has been constructed that describes the different decisions to be made and options provided, as well as their dependencies.

The complete XML representation of the “Serious Games” Decision Model can be found in Appendix A of this document. Later on (Chapter 6) we will provide the visualization of this model in the tool developed.

5 GuideaMaps: The iPad Application

We decided to call our iOS-based iPad application **GuideaMaps**, which is the concatenation of "Guided" "Ideas". This is because the tool can be considered as a guided ideation tool. The app will visualize an underlying Decision Model and allows constructing an Application Model.

The Decision Model will allow the app to display the choices and elements for the game requirement elicitation process. This model, which is defined in an XML format (see section 4.1), will be read by the program and could be visualized in an UML-like representation.

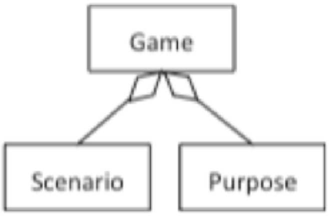


Figure-5.1 UML-like representation of features

The tool will however not present the model in strict "UML" notation (see figure-5.1) as this is too technical for a casual user and not all information and elements are relevant during every stage in the process. A more intuitive representation is needed.

Before making a final decision on the visualization, a number of paper mock-ups where created to validate some of the ideas and requirements.

As a Decision Model is hierarchical, we could use the iOS default way of presenting structured data. A list of top elements is presented from which the user can select one and drill-down to the next level where the parent's children are displayed. This top-down navigation is easy to implement using the iOS libraries and development frameworks but has some serious flaws:

- The user is navigating through a tree-based structure and constantly zooming in the different sub-features; he can easily loose track of the path that he has followed and a lot of clicks might be needed to backtrack.
- The user cannot see the relationship between different features as the representation is in a single dimension.
- Navigation to other parts of the feature model might require additional steps and blur the definition process of the serious game.

Therefore, an alternative representation should be used. Such an alternative representation could be based on the research done on available iOS applications (see section 3.1). Mind-mapping tools typically present information in a graphical tree-based structure with different "balloons" and "arrows" to define the links between the different elements.

As mentioned before, these existing tools don't provide the "dynamic guiding" concept that is required but some of the UI concepts can be used as a starting point.

The user can select the individual features and take immediate action on them. He can pinch to zoom in and out. He will be able to see the relationship instantly between the different features and optional "hide" information that is temporary not relevant.

By using colours and popovers ("dialog boxes" in iOS terminology), the user can provide comments for the chosen features and background information about the choices can be given.

In the next sections, we describe in more detail the visualization of the Decision Model (section 5.1), how the user is supported to create a consistent Application Model (section 5.2), how to deal with different Decision Models and Application Models (section 0), how to create new Application Model for a given Decision Model (section 5.4), how Application Models can be shared between users (section 5.5) and how new Decision Models can be loaded (section 0).

5.1 Visualization of the Decision (Feature) Model

5.1.1 Visualization of Features & Feature Composition

As our target audience does not consist of software engineers but of casual users, we do not use the term feature in our tool. We invented the term "guidea" (a portmanteau word for "guided" and "idea"), and because our representation is based on mind maps, de feature models are called "GuideaMaps". The application will display "**guideas**" as rectangles with rounded corners with different colours to represent different kind of guideas.

The top “compartment” of each rectangle displays the name of the guidea; the bottom one contains descriptive information that is provided by the decision model (e.g., an explanation of the guidea) or entered by the user (e.g., a description for the role of the guidea for the particular application) .

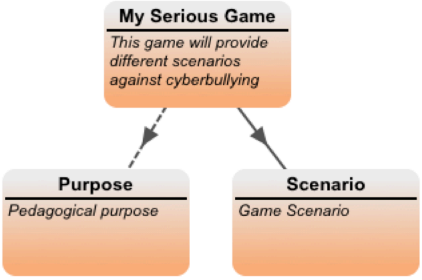

My Serious Game
This game will provide different scenarios against cyberbullying
Purpose
Pedagogical purpose
Scenario
Game Scenario

Figure-5.2 Guidea compartments

Figure-5.2 is visualizing the UML schema that was presented in figure-5.1 “UML-like representation of features”. The lines that connect the guideas will automatically be redrawn while dragging guideas.

The user will be able to select a guidea by tapping it with his finger and can drag it on the screen. By clicking outside the diagram and dragging onto the background, he can move the complete graph.

Figure-5.2 shows that “My Serious Game” is a **concrete feature** and is **decomposed into two different sub-features** (represented as sub-guideas): “Purpose” and “Scenario”.

Figure 5.3 shows that sub-guideas can be optional or mandatory; mandatory guideas are connected using a solid line; optional guideas are connected by a dotted line.

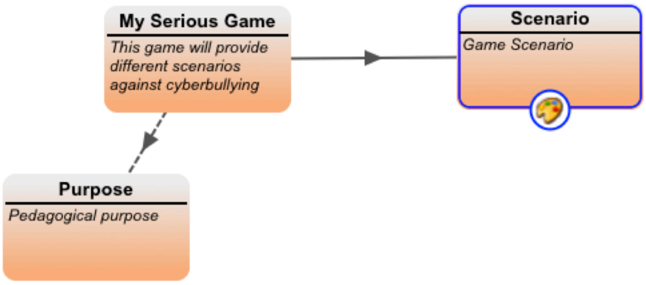

My Serious Game
This game will provide different scenarios against cyberbullying
Scenario
Game Scenario
Purpose
Pedagogical purpose

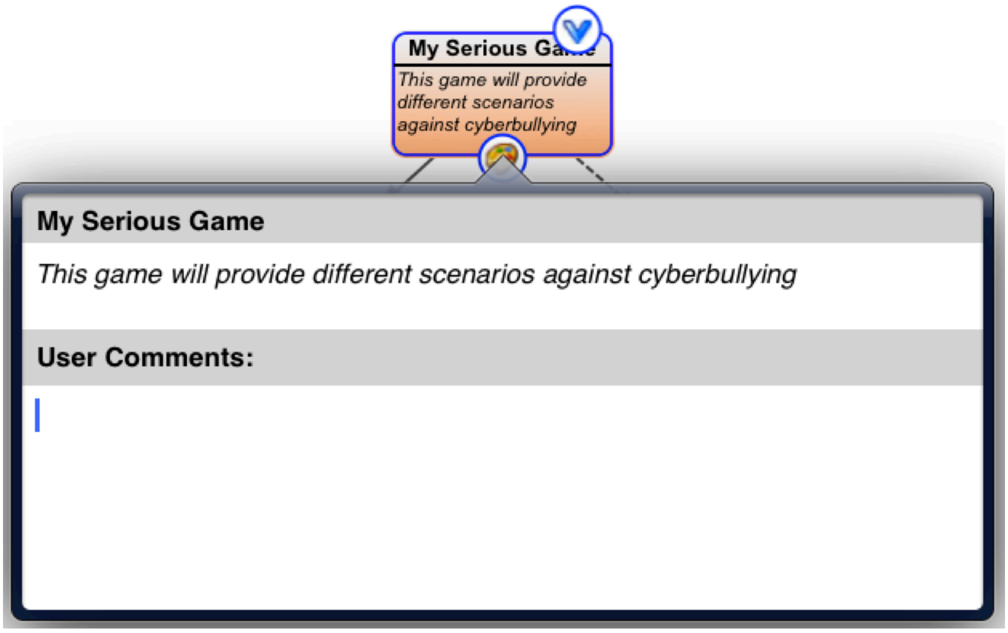
Figure-5.3 Feature with optional/mandatory sub-features

The user can provide a comment for each of the guideas; double tapping on a guidea will display the "User Comments" popover as displayed in figure-5.4.

The popover displays the name of the guidea and its description as defined in the Decision Model. The user can add a comment using the standard editing capabilities (auto-correct, copy/paste) of the device; an onscreen keyboard will popup and allow him to start typing.

The user can dismiss the popover by tapping outside the rectangle.

We have chosen for **context buttons on the boundaries of the selected guidea**; these buttons will only appear when the user taps on the guidea. We believe this gives a better user experience than providing an application menu on top of the screen to which the user must navigate to choose the appropriate action for the selected guidea.



My Serious Game
This game will provide different scenarios against cyberbullying
My Serious Game
This game will provide different scenarios against cyberbullying
User Comments:

Figure-5.4 User Comments for a selected Guidea

5.1.2 Visualization of Abstract Features

An abstract feature is a feature that is not concrete; it is a virtual feature that represents a **number of option feature(s)**. Abstract features are used to indicate variability and is represented by a dotted rectangle in a typical Feature Model (see figure-5.5).

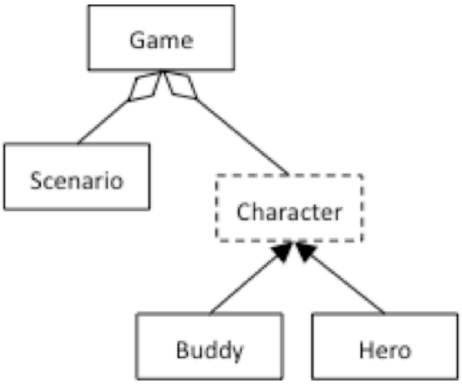


Figure-5.5 Abstract and concrete representation in UML

The app will display abstract features as "normal" guideas but with an extra compartment. The middle compartment will display the name of the abstract feature. As long as no option guidea has been selected, a number of question marks are displayed in the top compartment (see figure-5.6).

To indicate that the user has to take some action and select a concrete option guidea, it will be displayed in a different color (light yellow) (see figure-5.6).

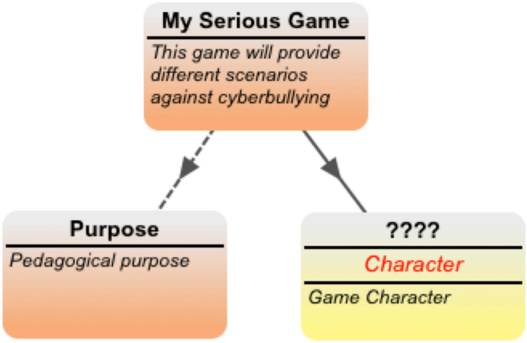


Figure-5.6 Abstract and concrete features as guideas

The user can tap on the abstract feature to select it; the “**Type**” button (question mark button) will be displayed to allow him to select an option for the guidea (see figure-5.7).

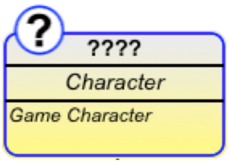


Figure-5.7 Abstract guidea with “Type” button

Tapping the “**Type**” button will bring up the “guideas” popover from which the user can select one of the **options** (see figure-5.8). The top row of the list will display the name and description of the abstract feature.

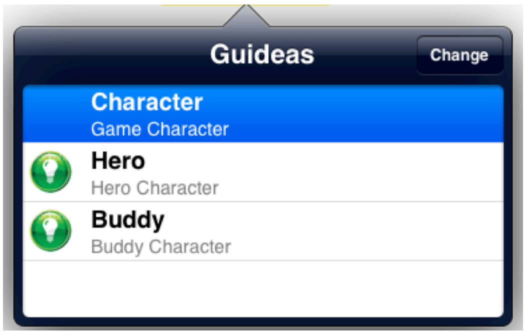


Figure-5.8 Guideas popover list

The next rows represent the list of options, i.e. the concrete guideas from which the user can choose. He can select a row and tap the “**Change**” button to effectuate the selection. This will immediately update the guidea with the user’s choice (see figure-5-9).

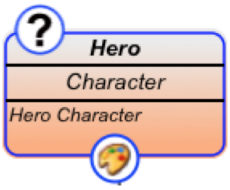


Figure-5.9 A selected concrete guidea

The user can still change his mind regarding the chosen guidea by tapping the “**Type**” button again and modify his choice.

Based on the cardinality of a guidea (see later), multiple choices can be made. The tool will prevent that the user can select the same guidea twice and will grey out the already selected guideas from the list (see figure-5.10).

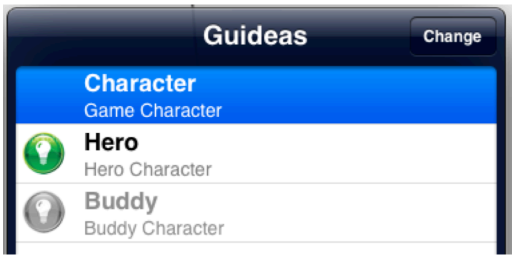


Figure-5.10 Greyed-out guideas in the list

5.1.3 Visualization of Decomposition of Abstract Features

An abstract feature can also be **decomposed into sub-features** as represented in the diagram of figure-5.11

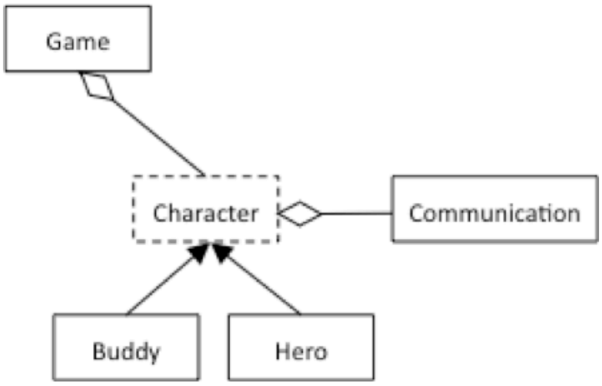


Figure-5.11 Abstract feature composition

GuideaMaps will visualize the defined composition and abstract feature functionality in a consistent and easy to understand representation.

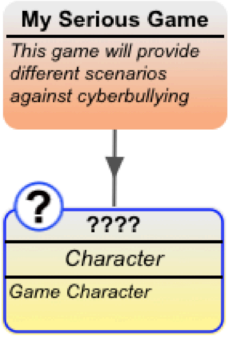

My Serious Game
This game will provide different scenarios against cyberbullying
?
????
Character
Game Character

Figure-5.12 Selected abstract guidea

After choosing an option guidea, by tapping the "Type" button" (see figure-5.12), the diagram will be updated and include the corresponding composing guideas (the resulting diagram is shown in figure-5.13).

Notice that for a guidea that contains “children” (sub-guideas), the **“Expand”/”Collapse”** button is displayed when selected. This allows the user to collapse part of the diagram to avoid too much screen cluttering.

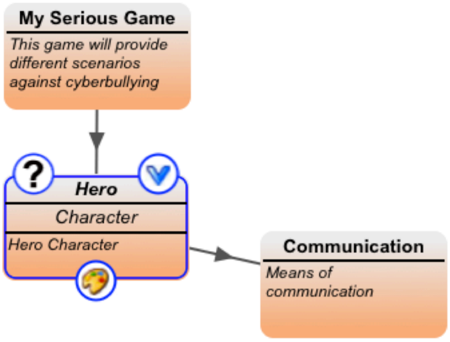


Figure-5.13 Concrete selected guidea

5.1.4 Visualization of Optional Features

The feature model can also include **“optional” sub-features** as part of the feature’s composition.

These features are typically represented by a dotted line between the parent feature and its optional sub-feature (see figure-5.14).

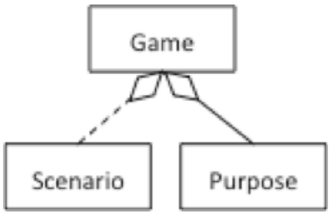


Figure-5.14 Optional sub-feature

These optional guideas are **not mandatory** and can be deselected by the user in the definition process.

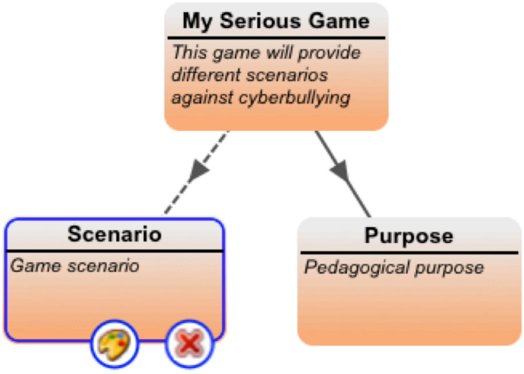

My Serious Game
This game will provide different scenarios against cyberbullying
Scenario
Game scenario
Purpose
Pedagogical purpose

Figure-5.15 Optional guidea

The iPad app will use the same convention as the UML representation; a dotted line will connect the optional guidea to its parent (see figure-5.15).

Optional guideas will include a "**Delete**" button (cross) when selected; tapping this button will disable the guidea (see figure-5.16).

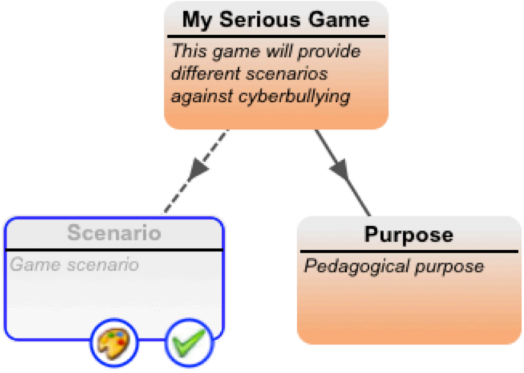

My Serious Game
This game will provide different scenarios against cyberbullying
Scenario
Game scenario
Purpose
Pedagogical purpose

Figure-5.16 Deleted optional guidea

Tapping the "**Enable**" button (checkmark) enable the guidea again.

5.1.5 Feature Cardinality

Feature cardinality is used to constraints the number of options that can be selected for an abstract feature. A cardinality of "1:1" which means that a parent feature is associated with minimum 1 and maximum 1 sub-feature. A "1:N" cardinality means that one parent feature can be linked with 1 to N sub-features of the given abstract type.

For optional relationships the cardinality can be expressed as "0:N". This means that 0 to N sub-features can be part of the parent feature. This implies that a child always has one parent.

A **"1:1" cardinality** indicates that one guidea is part of another parent feature and can be represented as "basic feature composition". This is visualized with a solid line between the features. **"0:1" cardinality** on the other hand, can be symbolized by an "optional feature", displayed by means of a dotted line between the parent and the child.

"1:N" feature cardinality allows the user to select 1 or more of the option features of the same abstract one (see figure-5.17).

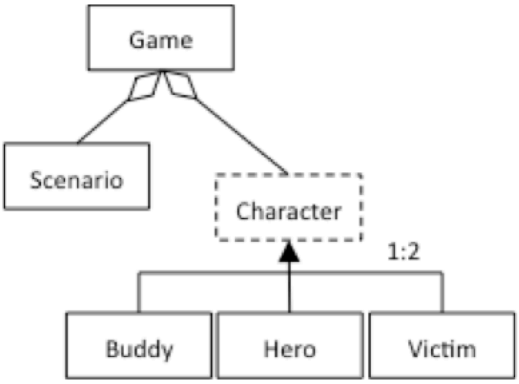


Figure-5.17 Abstract feature with cardinality

For 1:N feature cardinality, the tool will start with one "abstract" mandatory guidea for which an option can be selected. The "**Add**" button (plus) is displayed to allow adding additional guideas (see figure-5.18).

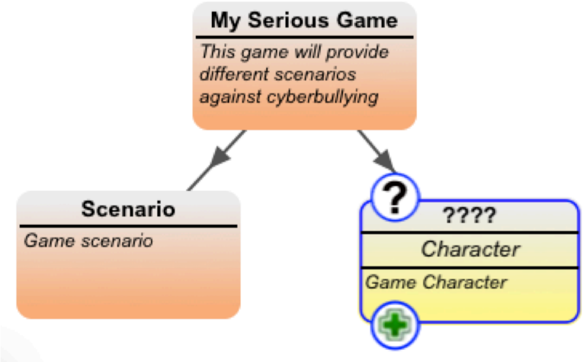

My Serious Game
This game will provide different scenarios against cyberbullying
Scenario
Game scenario
?
????
Character
Game Character

Figure-5.18 Abstract guidea with multiplicity

Clicking on the "**Add**" button (before or after selecting the option guidea) will display an additional guidea as shown in figure-5.19.

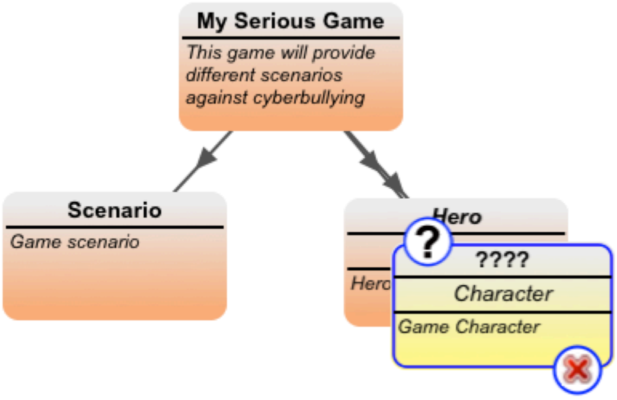


Figure-5.19 Multiple concrete guideas

"N:M" feature cardinality can be specified to define the minimum and maximum required choices.

The diagram in figure-5.20 represents a game that requires exactly 2 Character types. The tool will display 2 abstract "Character" guideas (with "?") (see figure-5.21) and the user can pick 1 of the 3 possible options for Character: Buddy, Hero or Victim.

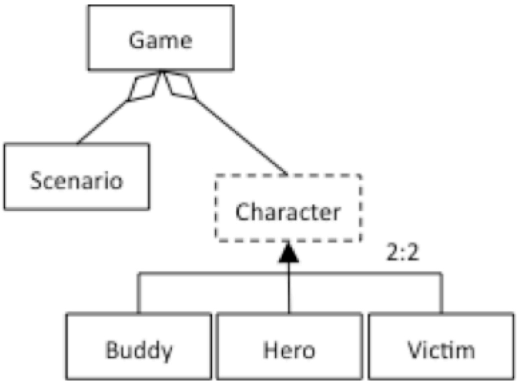


Figure-5.20 N:M cardinality feature

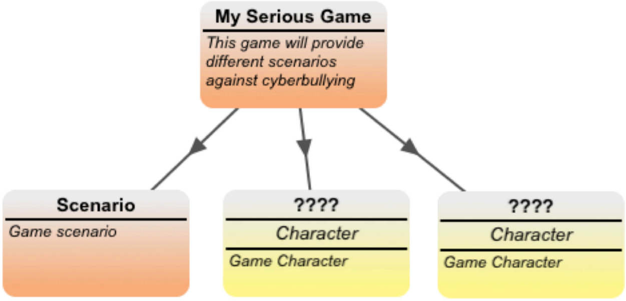


Figure-5.21 N:M abstract guideas

After this action, the user can select another character option for the second abstract "Character" feature.

"0:N" feature cardinality is implemented in a similar way as "1:N" (see for example figure-5.22); the first "abstract" feature will be marked as **optional** and can be **disabled** by the user.

The user can click the "**Add**" button to add additional guideas.

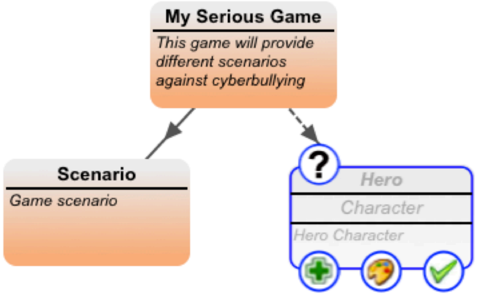

My Serious Game
This game will provide different scenarios against cyberbullying
Scenario
Game scenario
?
Hero
Character
Hero Character

Figure-5.22 Deleted optional guidea

5.2 Excluded / Required Guideas

As defined in the functional requirements, the tool should indicate the impact of the user's choices. Guideas may require and exclude others.

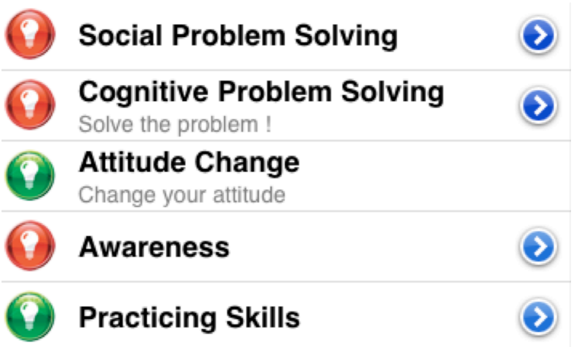
Based on these dependencies and on the guideas already selected, the tool will provide feedback on conflicts; possible conflicts and other required guideas in a visual way.

```
<feature id="goal" abstract="TRUE">
  <name>Pedagogical Goal</name>
  <description>Select the type of the pedagogical goal</description>
  <feature id="goal1" extends="TRUE">
    <name>Social Problem Solving</name>
    <requires>approach1</requires>
    <requires>context1</requires>
    <excludes>approach2</excludes>
    <excludes>approach3</excludes>
  </feature>
  <feature id="goal2" extends="TRUE">
    <name>Cognitive Problem Solving</name>
    <description>Solve the problem !</description>
    <requires>device2</requires>
    <excludes>context2</excludes>
  </feature>
```

Snippet-5.1 Excluded/Required Guideas

Based on the sample XML representation of Snippet-5.1 of the decision model, the user can select an option guidea for the abstract guidea from a list of option guideas.

For each option guidea in this list, the icon on the left will indicate if conflicts are detected; green means no conflicts discovered so far, red implies that a number of conflicts have been found (see figure 5.23).



Social Problem Solving

Cognitive Problem Solving
Solve the problem !

Attitude Change
Change your attitude

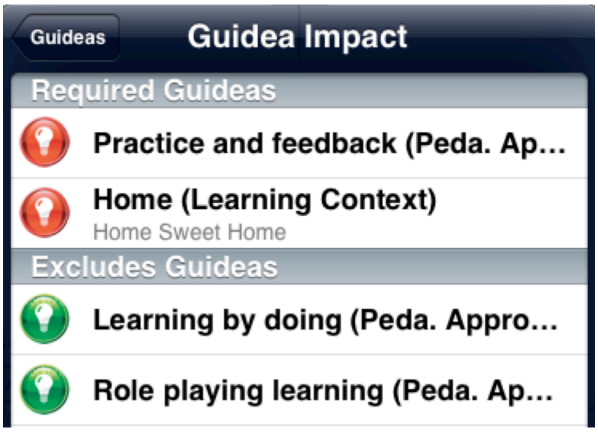
Awareness

Practicing Skills

Figure-5.23 Concrete Guideas with conflicts

When no icon is displayed, no "requires" or "excludes" have been defined in the decision model.

By clicking on the arrow on the right side of a row, the user can retrieve more information regarding the conflicts (in case of a red icon) or the potential conflicts (in case of a green icon) associated with this guidea (see figure 5.24).



Guidea Impact

Required Guideas

Practice and feedback (Peda. Ap...

Home (Learning Context)
Home Sweet Home

Excludes Guideas

Learning by doing (Peda. Appro...

Role playing learning (Peda. Ap...

Figure-5.24 Required and excluded Guideas

An overview of all the guideas that can have an impact (when the guidea is selected) will appear. The first list presents the required guideas, the second the excluded ones. Furthermore:

- A red icon indicates that the dependency is violated
- A green icon denotes that a dependency is defined in the decision model but not violated.

Guideas that are in conflict with some dependencies will be shown with a red title and red dotted outer frame in the diagram (see figure-5.25).

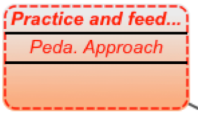


Figure-5.25 Guidea in conflict

5.3 Switching between GuideaMaps

As defined in the initial requirements, the user should be able to switch between different requirements documents (i.e. GuideaMaps) that he has been working on. This can be accomplished by pressing the “**GuideaMaps**” button on the top of the screen.

This will bring up the “**GuideaMaps Browser**” popover (see figure-5.26). In this popover, the user can pick a different GuideaMap or perform a number of related actions.

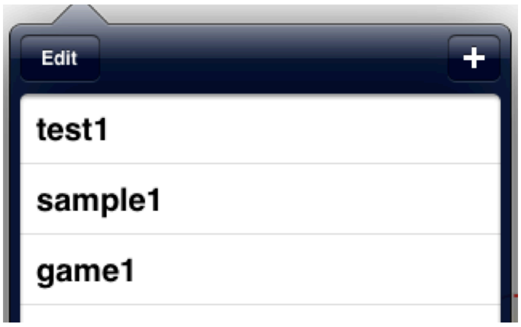


Figure-5.26 GuideaMaps Browser

The user can delete GuideaMaps from the list (which will be removed from the device) by clicking the “**Edit**” button. A small “Removal” icon is displayed to prepare a GuideaMap for deletion (see figure 5.27).

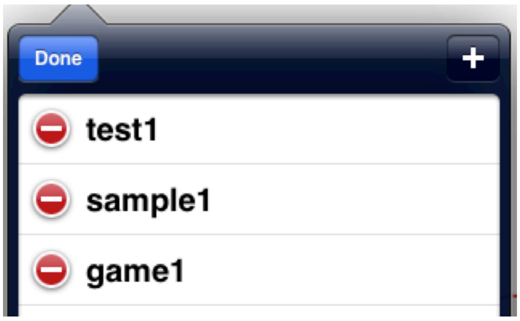


Figure-5.27 Preparing GuideaMap documents for deletion

Clicking the icon will display the “**Delete**” button; pressing this button will physically remove the GuideaMap from the system (see figure-5.28).

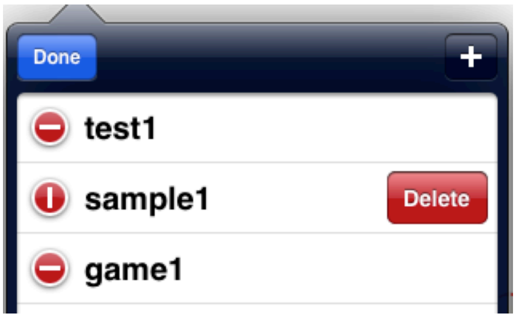


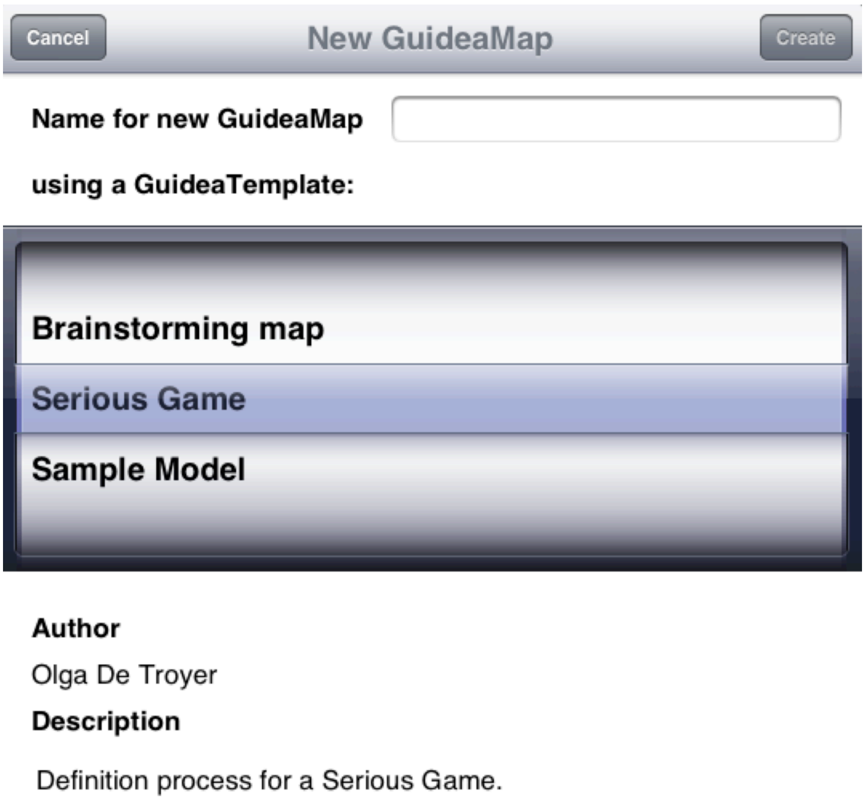
Figure-5.28 Deleting a GuideaMap document

NOTE:
GuideaMap files are auto-saved by the system; the user doesn't have to manually save the document to persist them on the device. Whenever changes are detected, the system will write them asynchronously to the device's file system.

5.4 Creating New GuideaMaps

The user can create new GuideaMaps, based on a selected **GuideaTemplate**. A GuideaTemplate is the name used in the tool for a Decision Model represented in XML format. The product will be pre-packaged with a number of templates; the user can load additional ones.

Clicking on the "**GuideaMaps**" button on the top of the screen will display the "**GuideaMaps Browser**" popover. When the user clicks the "**+**" button, the new GuideaMaps sheet is presented (see figure-5.29).



Name for new GuideaMap

using a GuideaTemplate:

Brainstorming map
Serious Game
Sample Model

Author
Olga De Troyer
Description
Definition process for a Serious Game.

Figure-5.29 Adding a new GuideaMap document

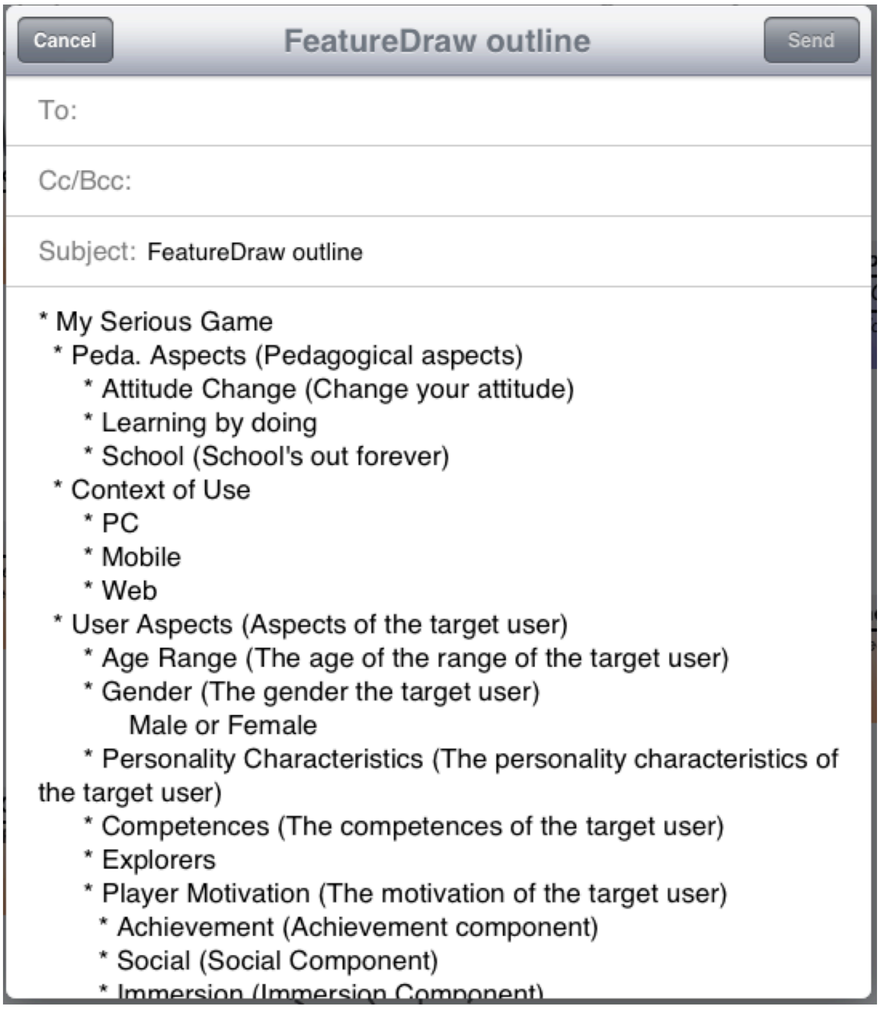
In this sheet, the user can pick one of the available GuideaTemplates (by scrolling through the picker). While scrolling, the author and description about the template will be presented. After choosing the appropriate GuideaTemplate and entering a new name for the new GuideaMap, the user can press the "**Create**" button to physically create the file on the device and load the corresponding GuideaTemplate into memory.

The newly created GuideaMap will be added in the list of the “GuideaMap Browser”.

5.5 Sharing GuideaMaps

iOS provides a sandboxed environment in which documents are stored together with the application. Opening and sharing documents between applications is strictly managed and limited by the OS.

The user can share his GuideaMaps with others by using the “**Email**” button on top of the screen. This will capture the current state of the GuideaMap (including the comments that were provided by the user) into one concise email message that can be sent out (see figure-5.30 for an example).



To:

Cc/Bcc:

Subject: FeatureDraw outline

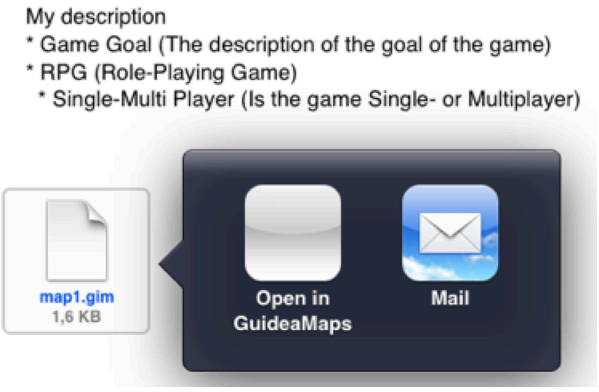
```
* My Serious Game
  * Peda. Aspects (Pedagogical aspects)
    * Attitude Change (Change your attitude)
    * Learning by doing
    * School (School's out forever)
  * Context of Use
    * PC
    * Mobile
    * Web
  * User Aspects (Aspects of the target user)
    * Age Range (The age of the range of the target user)
    * Gender (The gender the target user)
        Male or Female
    * Personality Characteristics (The personality characteristics of
the target user)
    * Competences (The competences of the target user)
    * Explorers
    * Player Motivation (The motivation of the target user)
      * Achievement (Achievement component)
      * Social (Social Component)
      * Immersion (Immersion Component)
```

Figure-5.30 Sending a GuideaMap document

The user can tap the To: or Cc/Bcc: fields and add email addresses. When finished, he can click the “**Send**” button to initiate the email process.

The GuideaMap document itself (using .gim extension) will be appended to the end of the mail message. This allows the user to communicate the GuideMap to another user’s device.

When a user receives a GuideaMap (with .gim extension) in his mailbox, he can open the document and launch the GuideaMaps application by pressing the document for a few seconds until the application popover is displayed (see figure-5.31).



My description
* Game Goal (The description of the goal of the game)
* RPG (Role-Playing Game)
  * Single-Multi Player (Is the game Single- or Multiplayer)

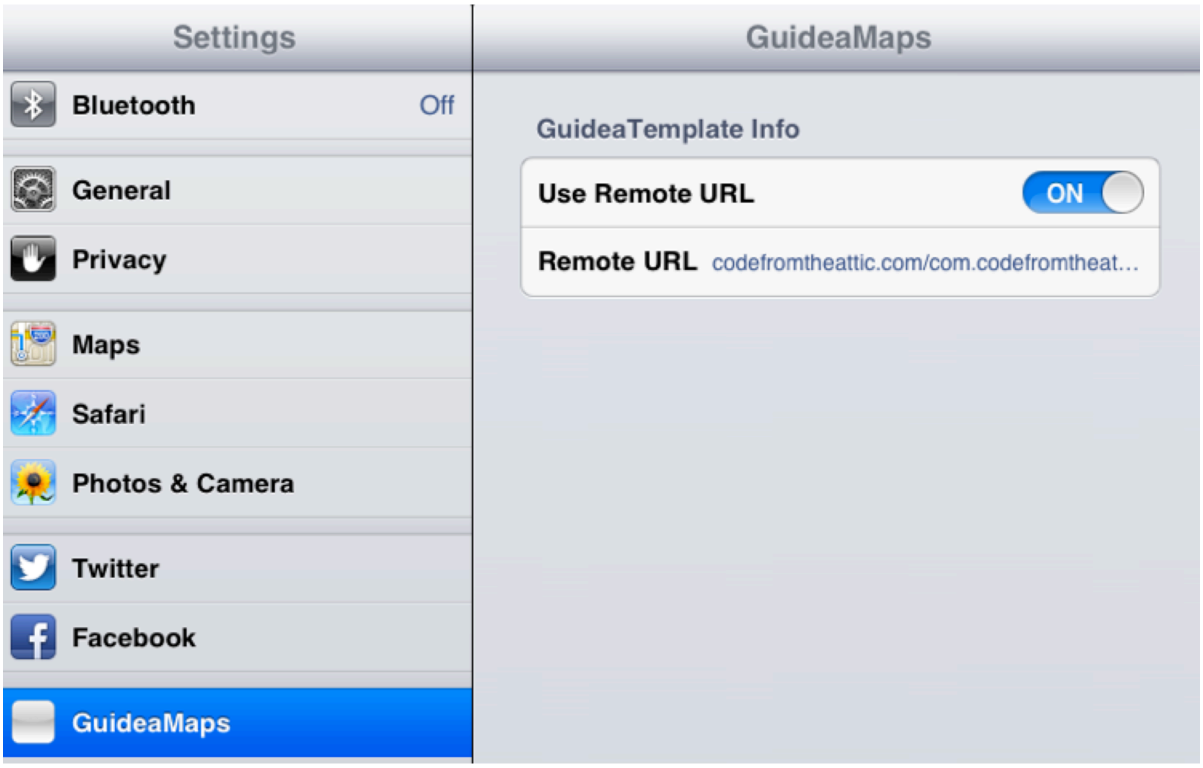
Figure-5.31 Opening a GuideaMap document

Selecting “**Open in GuideaMaps**” will start the application and display the content of the document.

Integration with Apple’s iCloud or Dropbox could be implemented in future versions.

5.6 Loading GuideaTemplates

The user can load additional GuideaTemplates (i.e. Decision Models) using the web. To do this, he should launch the iOS "**Settings**" application and click on the GuideaMaps app (see figure-5.32).



GuideaTemplate Info

Use Remote URL: ON

Remote URL: codefromtheattic.com/com.codefromtheat...

Figure-5.32 Loading additional GuideaTemplates

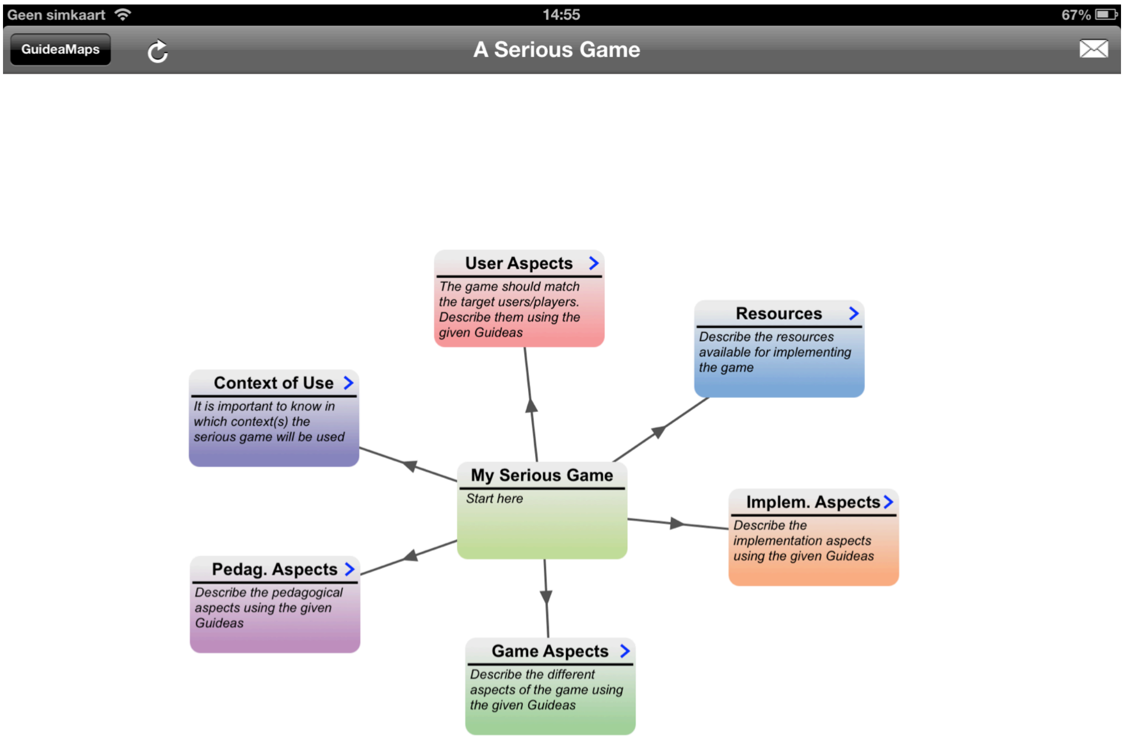
This will display the GuideaMaps specific settings:

- The "**Use Remote URL**" should be enabled to allow the application to connect to the Internet to load remote GuideaTemplates.
- In the "**Remote URL**" field the location of the GuideaTemplate must be specified. It is recommended to include the inverse domain name in the name of the GuideaTemplate to avoid "name clashes" between different templates (e.g. com.codefromtheattic.games.xml).

NOTE:
The remote GuideaTemplate is only retrieved when the GuideaMaps app is initiated; the application will typically remain active for a longer period of time to allow the user to switch between different apps.

6 Demonstration: My Serious Game

Based on the experiences acquired in the Friendly ATTAC project, a Decision Model for serious games have been developed. The Decision Model covers different aspects to be consider during the requirement elicitation process: the context of use; user aspects; pedagogical aspects; available resources; the game aspects; and different implementation aspects. Each of these aspects is modelled as a sub-model. The GuideaMap corresponding with this Decision Model (given in XML format in Appendix A) is shown in figure-6.1. In this figure, all sub-models are collapsed.



User Aspects: The game should match the target users/players. Describe them using the given Guideas

Resources: Describe the resources available for implementing the game

Context of Use: It is important to know in which context(s) the serious game will be used

My Serious Game: Start here

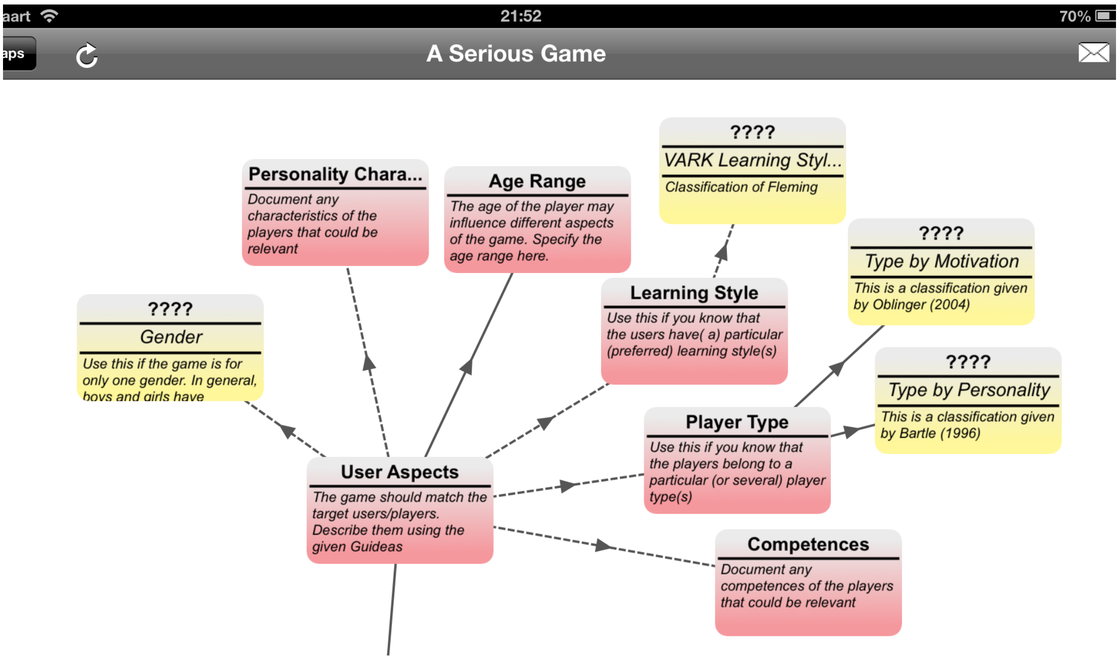
Implem. Aspects: Describe the implementation aspects using the given Guideas

Pedag. Aspects: Describe the pedagogical aspects using the given Guideas

Game Aspects: Describe the different aspects of the game using the given Guideas

Figure-6.1 "My Serious Game" in GuideMaps

Figure-6.2 to figure-6.7 shows the different sub-models. In each figure one of the sub-models is shown; the others are collapsed.



????
VARK Learning Styl...
Classification of Fleming

Personality Chara...
Document any characteristics of the players that could be relevant

Age Range
The age of the player may influence different aspects of the game. Specify the age range here.

????
Type by Motivation
This is a classification given by Oblinger (2004)

Learning Style
Use this if you know that the users have(a) particular (preferred) learning style(s)

????
Gender
Use this if the game is for only one gender. In general, boys and girls have

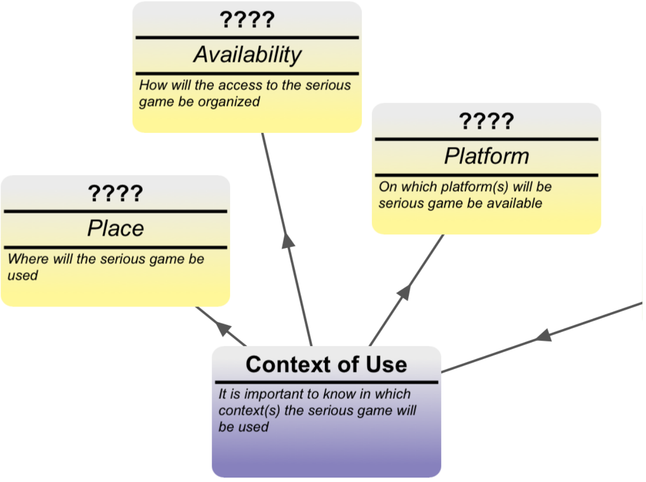
????
Type by Personality
This is a classification given by Bartle (1996)

Player Type
Use this if you know that the players belong to a particular (or several) player type(s)

User Aspects
The game should match the target users/players. Describe them using the given Guideas

Competences
Document any competences of the players that could be relevant

Figure-6.2 "User Aspects" sub-model of My Serious Game

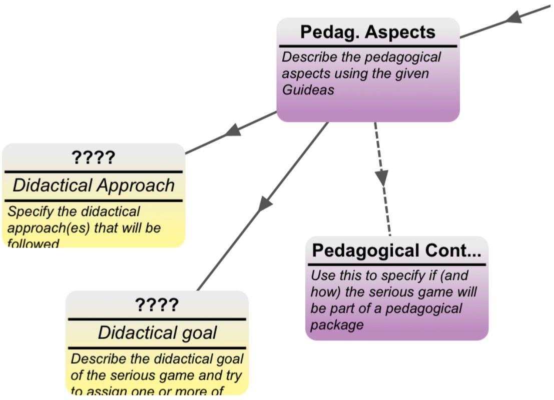

????
Availability
How will the access to the serious game be organized

????
Platform
On which platform(s) will be serious game be available

????
Place
Where will the serious game be used

Context of Use
It is important to know in which context(s) the serious game will be used

Figure-6.3 "Context of Use" sub-model of My Serious Game


Pedag. Aspects
Describe the pedagogical aspects using the given Guideas

????
Didactical Approach
Specify the didactical approach(es) that will be followed

Pedagogical Cont...
Use this to specify if (and how) the serious game will be part of a pedagogical package

????
Didactical goal
Describe the didactical goal of the serious game and try to assign one or more of

Figure-6.4 "Pedagogical Aspects" sub-model of My Serious Game

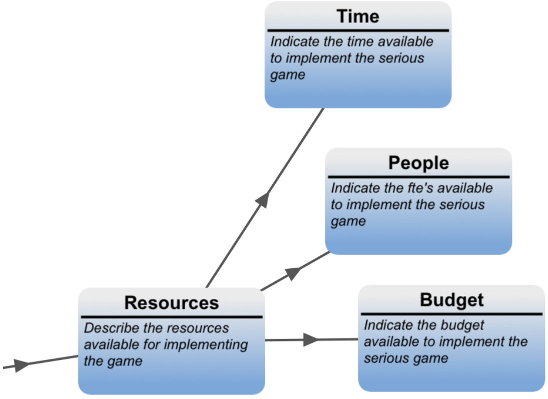
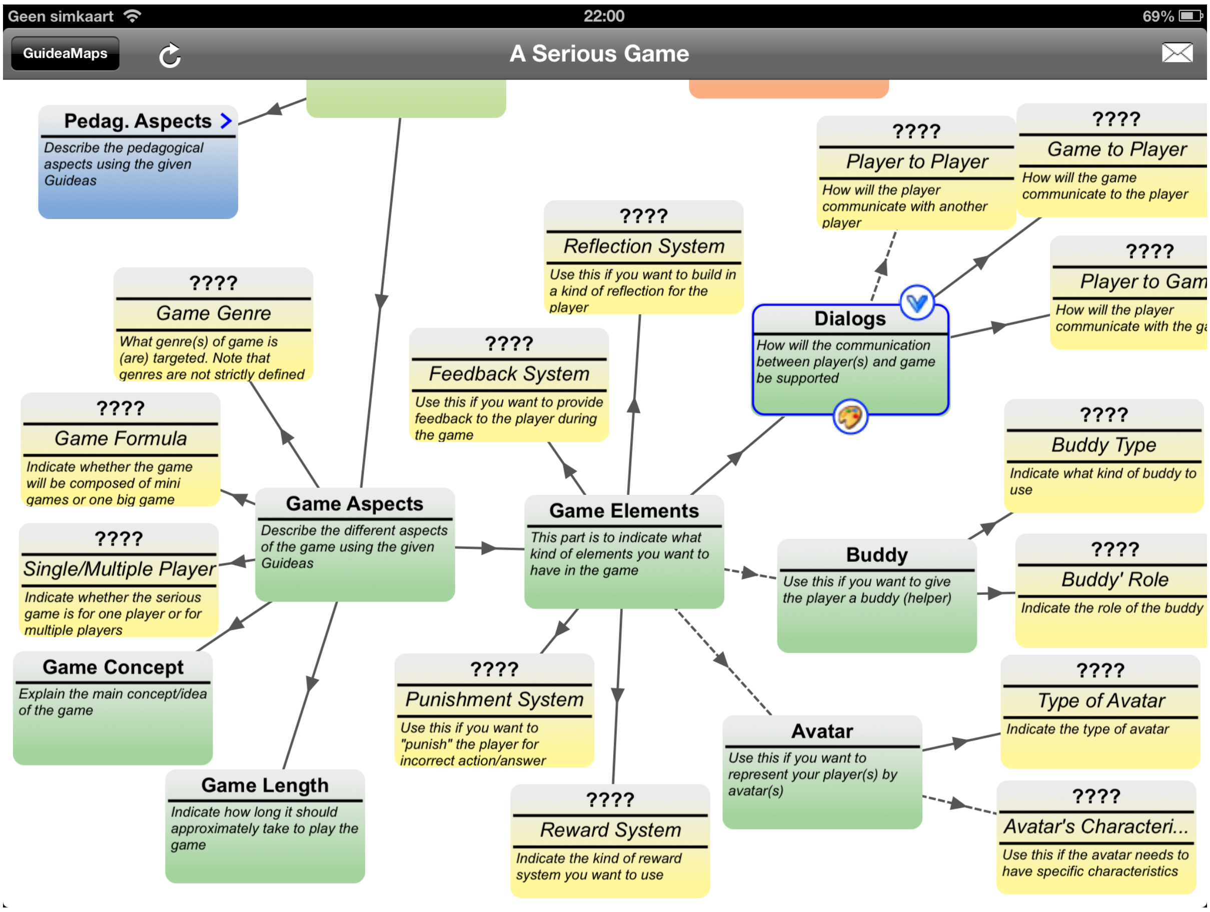

Time
Indicate the time available to implement the serious game
People
Indicate the fte's available to implement the serious game
Resources
Describe the resources available for implementing the game
Budget
Indicate the budget available to implement the serious game

Figure-6.5 "Resources" sub-model of My Serious Game



Pedag. Aspects
Describe the pedagogical aspects using the given Guideas

????
Game Genre
What genre(s) of game is (are) targeted. Note that genres are not strictly defined

????
Game Formula
Indicate whether the game will be composed of mini games or one big game

????
Single/Multiple Player
Indicate whether the serious game is for one player or for multiple players

Game Concept
Explain the main concept/idea of the game

Game Aspects
Describe the different aspects of the game using the given Guideas

Game Length
Indicate how long it should approximately take to play the game

????
Reflection System
Use this if you want to build in a kind of reflection for the player

????
Feedback System
Use this if you want to provide feedback to the player during the game

Game Elements
This part is to indicate what kind of elements you want to have in the game

????
Punishment System
Use this if you want to "punish" the player for incorrect action/answer

????
Reward System
Indicate the kind of reward system you want to use

????
Player to Player
How will the player communicate with another player

????
Game to Player
How will the game communicate to the player

????
Player to Gam
How will the player communicate with the ga

Dialogs
How will the communication between player(s) and game be supported

????
Buddy Type
Indicate what kind of buddy to use

Buddy
Use this if you want to give the player a buddy (helper)

????
Buddy' Role
Indicate the role of the buddy

Avatar
Use this if you want to represent your player(s) by avatar(s)

????
Type of Avatar
Indicate the type of avatar

????
Avatar's Characteri...
Use this if the avatar needs to have specific characteristics

Figure-6.6 "Game Aspects" sub-model of My Serious Game

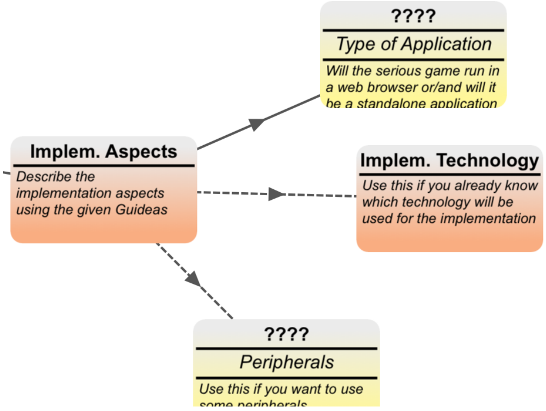
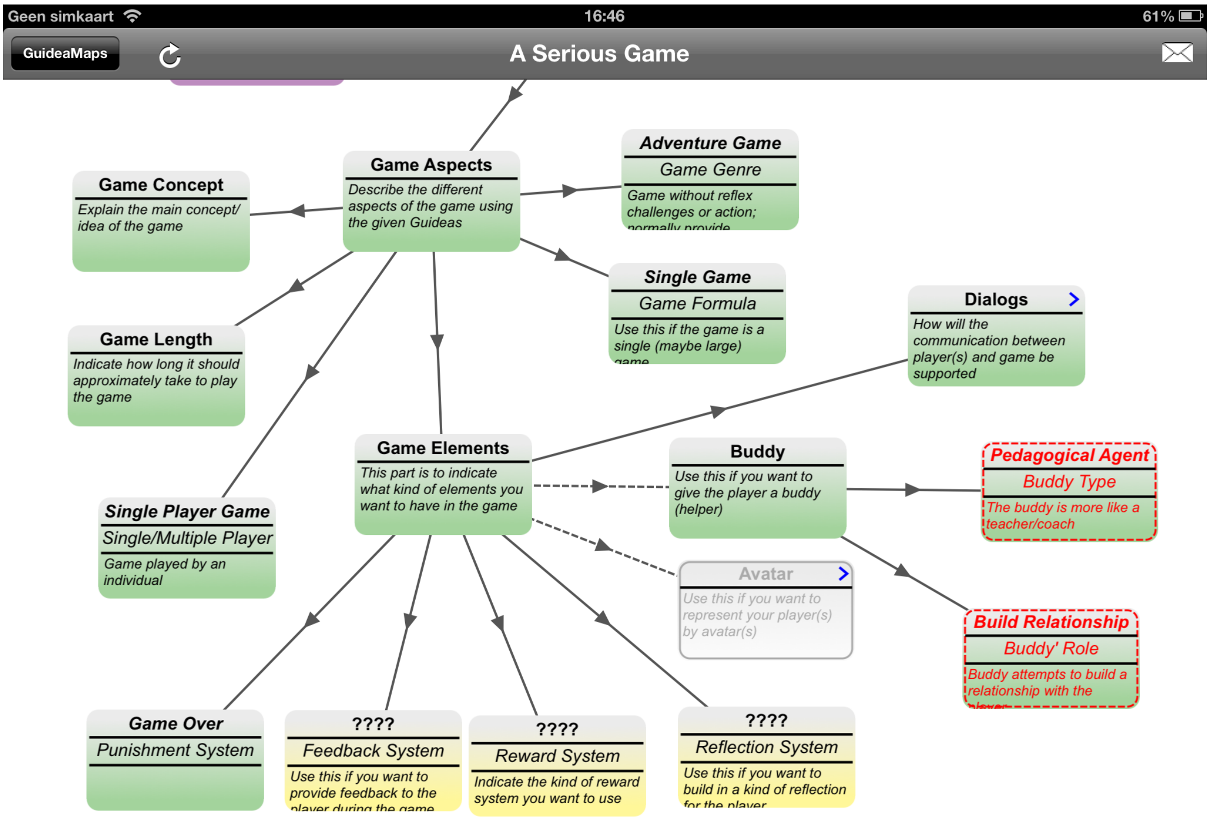

????
Type of Application
Will the serious game run in a web browser or/and will it be a standalone application
Implem. Aspects
Describe the implementation aspects using the given Guideas
Implem. Technology
Use this if you already know which technology will be used for the implementation
????
Peripherals
Use this if you want to use some peripherals

Figure-6.7 "Implementation Aspects" sub-model of My Serious Game

Figure-6.8 shows a (part of a) partially filled Application Model for the My Serious Game Decision Model. One can notice that the optional Avatar has been deselected (the guidea is grey); for Buddy Type and Buddy's Role options have been selected but these currently result in some conflicts (indicated by the red dotted borders and red text); For Feedback System, Reward System, and Reflection Systems options still need to be selected (indicated by the "???" and the yellow color).



Adventure Game
Game Genre
Game without reflex challenges or action; normally provide

Game Concept
Explain the main concept/ idea of the game

Game Aspects
Describe the different aspects of the game using the given Guideas

Single Game
Game Formula
Use this if the game is a single (maybe large) game

Dialogs
How will the communication between player(s) and game be supported

Game Length
Indicate how long it should approximately take to play the game

Game Elements
This part is to indicate what kind of elements you want to have in the game

Buddy
Use this if you want to give the player a buddy (helper)

Pedagogical Agent
Buddy Type
The buddy is more like a teacher/coach

Single Player Game
Single/Multiple Player
Game played by an individual

Avatar
Use this if you want to represent your player(s) by avatar(s)

Build Relationship
Buddy' Role
Buddy attempts to build a relationship with the player

Game Over
Punishment System

????
Feedback System
Use this if you want to provide feedback to the player during the game

????
Reward System
Indicate the kind of reward system you want to use

????
Reflection System
Use this if you want to build in a kind of reflection for the player

Figure-6.8 Partially filled-in GuideaMap for My Serious Game

7 Implementation Details

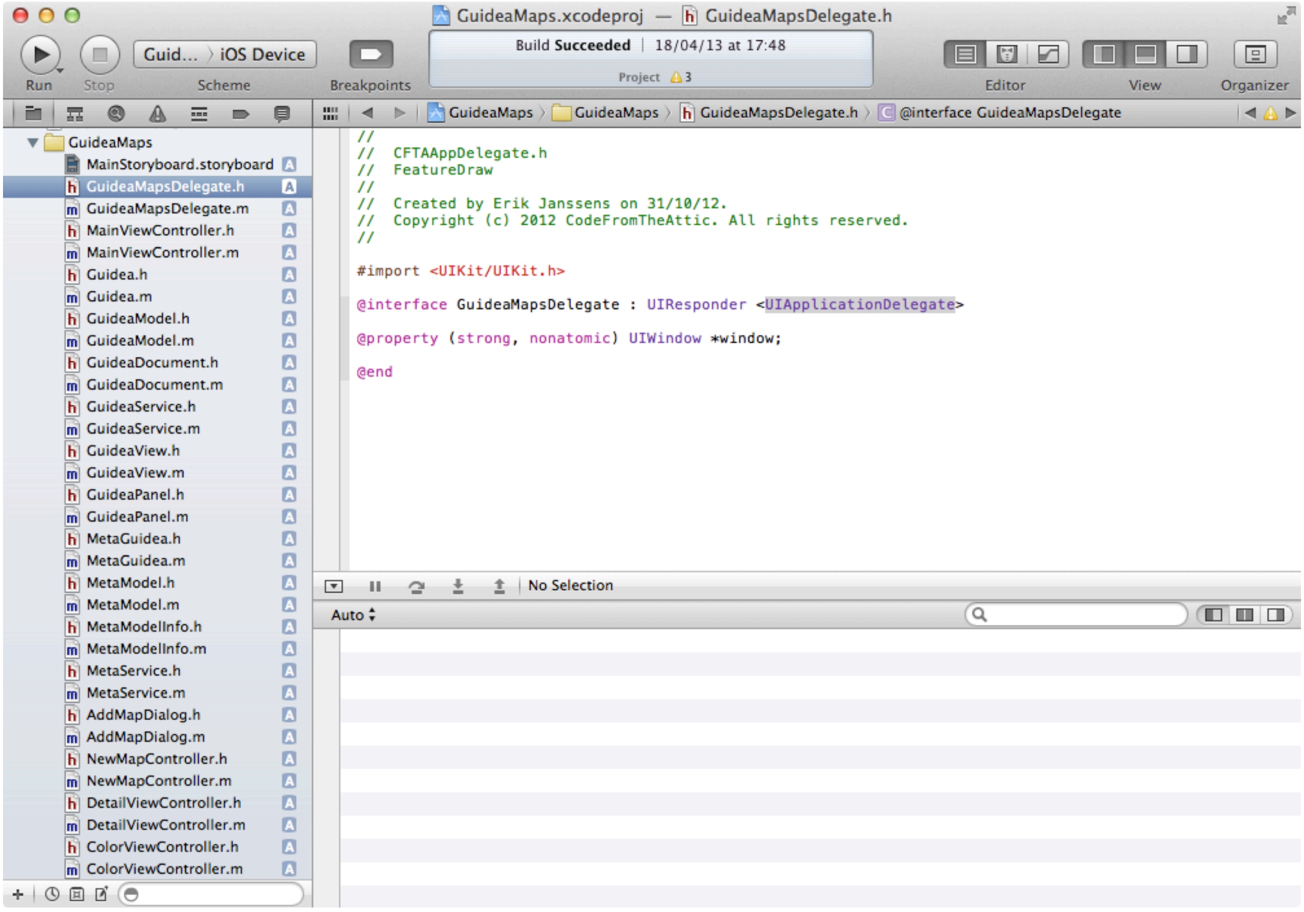
In this chapter, we provide more in-depth information about the implementation of the iPad application. In the first part (7.1 "Working with iOS") we describes the tools, language and patterns that are used when building a typical iOS application. In the second part (7.2 "High level structure of the application") we discuss the structure of the GuideaMaps application: its controllers, storyboards, and domain model. The last part (7.3 "Distributing the Software") provides info on different distribution scenarios.

7.1 Working with iOS

7.1.1 Setting Up the Development Environment

The first thing an iOS developer needs to install is Apple's iOS software development kit (SDK) and Xcode (Apple's integrated development environment). Xcode provides everything the developer needs to create applications for the iPhone and iPad.

It contains an advanced source editor, a graphical user interface editor and many other powerful features. The IDE works in a single "workspace" window (see



Build Succeeded | 18/04/13 at 17:48

Project ⚠3

```
//
//  CFTAAppDelegate.h
//  FeatureDraw
//
//  Created by Erik Janssens on 31/10/12.
//  Copyright (c) 2012 CodeFromTheAttic. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface GuideaMapsDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@end
```

No Selection

Figure-7.1 Xcode Workspace window

figure 7.1) that includes most of the tools you need to develop applications. Within this window the developer can switch between writing code, debugging and designing the user interface.

The iOS SDK extends the Xcode toolset to include the tools, compilers, and frameworks you need specifically for a given version of iOS (e.g. iOS 6.1.0).

To start working with iOS the following steps should be taken:

- Download the latest version of Xcode.
Use the Mac App Store app on your Mac to download the latest version of Xcode (freely available). The downloaded Xcode application, includes the iOS SDK.
(Note: The Mac App Store app is installed with Mac OS X version 10.7 and later)

- Enroll as an Apple Developer in the iOS Developer Program (99€/year).
If you want to test apps on different iOS devices and potentially distribute them, you need to enroll in this program. Once enrolled, the developer gets full access to the iOS Dev Center and the iOS Provisioning Portal (to set-up certificates, register iOS devices and upload to the App Store).

7.1.2 Objective-C[19]

Objective-C is a simple and elegant object-oriented language that powers all iOS apps.

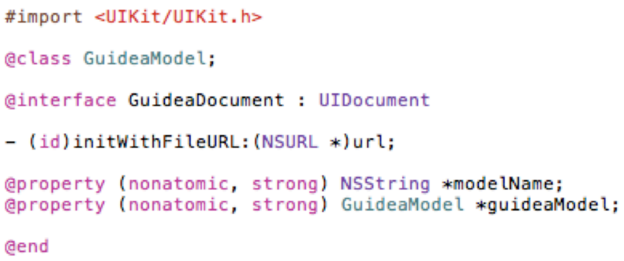
The Objective-C language provides the syntax for defining classes and methods, for calling methods of objects and for dynamically extending classes and creating programming interfaces adapted to address specific problems. It is a superset of the C programming language and supports the same basic syntax as C. The developer can use the familiar elements of C such as primitive types (e.g., int, float), structures, functions, pointers and control-flow constructs such as if...else and for statements. He also has access to the standard C library routines (e.g., stdlib.h and stdio.h).

Objective-C adds a number of syntax changes and features to ANSI C:

- Definition of new classes (see figure-7.2)
- Class and instance methods
- Method invocation (called messaging)
- Declaration of properties (and automatic synthesizing of accessor methods)
- Static and dynamic typing
- Blocks—encapsulated segments of code that can be executed at any time
- Extensions to the base language such as protocols and categories

[19] http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/Write Objective-CCode/Write Objective-CCode/WriteObjective-CCode.html

Objective-C is a dynamic language and provides most of the abstractions and mechanisms that can be found in other object-oriented languages. It is dynamic in that it permits an app's behavior to be determined when it is running (at runtime) rather than being fixed when the app is built.



```
#import <UIKit/UIKit.h>

@class GuideaModel;

@interface GuideaDocument : UIDocument

- (id)initWithFileURL:(NSURL *)url;

@property (nonatomic, strong) NSString *modelName;
@property (nonatomic, strong) GuideaModel *guideaModel;

@end
```

Figure-7.2 Objective-C class definition

7.1.3 Frameworks[20]

The developer writes the code that is specific for his application. He combines his source code with the frameworks that are provided by Apple (as part of the iOS SDK). These frameworks contain a library of classes and methods that can be used in the application. The framework libraries are included in the iOS on the device and can be shared by more than one app at the same time.

A custom application typically links to multiple frameworks using a well-defined "application-programming interface" (API). For each framework, a header file is provided that specifies the available classes, data structures and protocols. By reusing these frameworks, the developer can save time and effort; the code is efficient and secure. Apple also provides system frameworks to access the

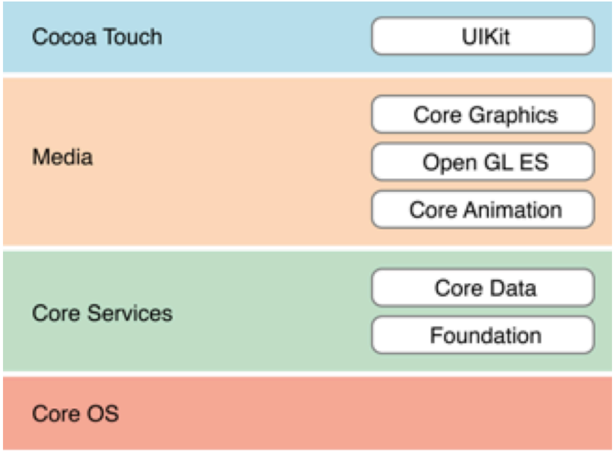


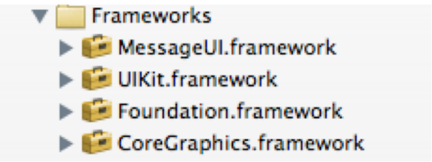
Figure-7.3 The iOS layers and key frameworks

[20] http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/SurveytheMajorFrameworks /SurveytheMajorFrameworks/SurveytheMajorFrameworks.html

underlying hardware.

Every framework belongs to a specific layer of the iOS system. Each layer builds on the ones below it (see figure-7.3). Use higher-level frameworks instead of lower-level frameworks whenever possible; higher-level frameworks provide object-oriented abstractions for lower-level constructs.

To use one or more frameworks, add it to the project so that the application can link to it. Most apps link to the Foundation, UIKit, and Core Graphics frameworks. The developer can always add additional frameworks to his project if the core set of frameworks does not meet his app's requirements. Figure-7.4 shows the frameworks used for the GuideaMaps app.



▼ Frameworks
- ▶ MessageUI.framework
- ▶ UIKit.framework
- ▶ Foundation.framework
- ▶ CoreGraphics.framework

Figure-7.4 Frameworks used in GuideaMaps project

7.1.4 Design Patterns

"A design pattern (Gamma, Helm, Johnson, & Vlissides, 1994) is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Patterns are formalized best practices that the programmer must implement themselves in the application." [21]

A number of design patterns are used in the app. We briefly describe them.

Model-View-Controller[22]

The Model-View-Controller design pattern (MVC) assigns objects in an application one of three roles: model, view, or controller. The pattern defines the way objects communicate with each other (see figure-7.5). Each of the three types of objects is separated from the others by abstract boundaries and communicates with objects of the other types across those boundaries. The collection of objects of a certain MVC type in an application is sometimes referred to as a layer—for example, a model layer.

[21] http://en.wikipedia.org/wiki/Software_design_pattern

[22] http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/StreamlineYourAppswithDes ignPatterns/StreamlineYourApps/StreamlineYourApps.html

MVC is central to a good design for any iOS app or Mac app. Apps having an MVC design are also more easily extensible than other apps.

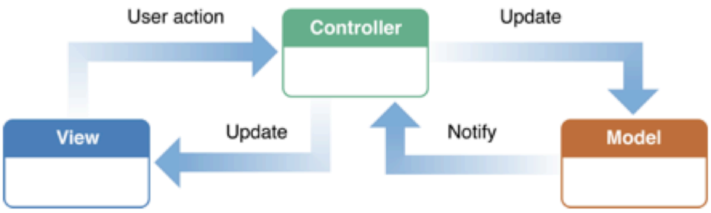


Figure-7.5 MVC design pattern

Delegation

In delegation, an object called the delegate acts on behalf of, and at the request of, another object. That other, delegating, object is typically a framework object. As shown in figure-7.6, at some point in execution, a message is send to its delegate; the message tells the delegate that some event is about to happen and asks for some response.

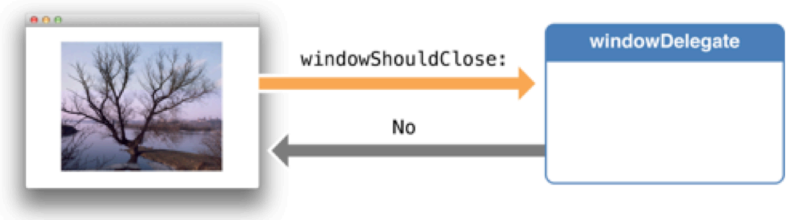


Figure-7.6 Delegation design Pattern

Delegation is thus a means for injecting application-specific behavior into the workings of a framework class—without having to subclass that class. It is a common and powerful design for extending and influencing the behavior of the frameworks.

```
   @implementation GuideaMapsDelegate

   - (BOOL)application:(UIApplication *)application
       didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
   {
       NSURL *url = (NSURL *)[launchOptions
   valueForKey:UIApplicationLaunchOptionsURLKey];
       if (url != nil && [url isFileURL])
           …
       return YES;
   }

   - (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
       sourceApplication:(NSString *)sourceApplication
   annotation:(id)annotation
   {
       if (url != nil && [url isFileURL]) {
           UIWindow *window = [self window];
```

```
            MainViewController *controller = (MainViewController * )
               window.rootViewController;
            [controller openDocumentFromMail:url];
            …
        }
        return YES;
    }
```

Snippet-7.1 GuideaMapsDelegate methods

Snippet-7.1 illustrates some of the key methods of the `GuideaMapsDelegate` class. The framework will notify the delegate that the launch process is almost done and the app is almost ready to run. It will also ask the delegate to open a resource identified by URL.

Protocols

In Objective-C protocols are used to declare a list of methods that are used or may be implemented in any other class. Protocols are declared with the @protocol directive. They have no curly brackets with variables since they cannot have any variables associated with them. An example is shown in snippet-7.2.

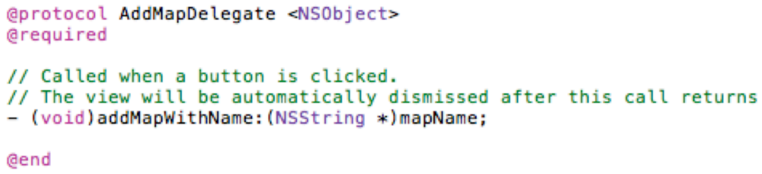
```
@protocol MyProtocol
- (void) thisMethodIsRequired;

@optional
- (void) thisMethodIsOptional;

@required
- (void) thisIsARequiredMethod;
@end
```

Snippet-7.2 Protocol example

Protocol method can be marked as required or optional to be implemented. Required methods are marked with the @required keyword, and optional methods are marked with the @optional keyword. A protocol definition used in GuideaMaps is shown in figure-7.7.



```
@protocol AddMapDelegate <NSObject>
@required

// Called when a button is clicked.
// The view will be automatically dismissed after this call returns
- (void)addMapWithName:(NSString *)mapName;

@end
```

Figure-7.7 @protocol definition in GuideaMaps

7.2 High level Structure of the Application

7.2.1 Starting the Application

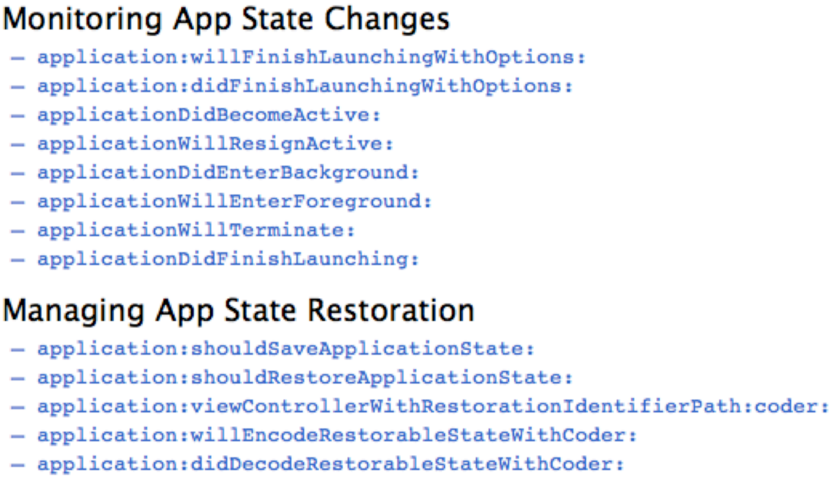
As with all C-based applications, the entry point of the program is the main procedure (see snippet-7.3).

```
int main(int argc, char *argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
            NSStringFromClass([GuideaMapsDelegate class]));
      }
  }
```

Snippet-7.3 main function

The **UIApplicationMain** function instantiates the application object from the principal class and creates the proper delegate (if any) from the given class. It also sets the delegate for the application and the main event loop so that it begins processing events. If the application has a storyboard configured (see section 7.2.3), it will load it and instantiate the viewController that is marked as the "Initial View Controller".

The **UIApplicationDelegate** protocol (see figure-7.8) declares methods that are implemented by the delegate of the singleton **UIApplication** object. These methods provide information about key events in an application's execution such as when it finished launching, when it is about to be terminated, when memory is low, and when important changes occur. Implementing these methods gives the developer a chance to respond to these system events and respond appropriately.



Monitoring App State Changes

- application:willFinishLaunchingWithOptions:
- application:didFinishLaunchingWithOptions:
- applicationDidBecomeActive:
- applicationWillResignActive:
- applicationDidEnterBackground:
- applicationWillEnterForeground:
- applicationWillTerminate:
- applicationDidFinishLaunching:

Managing App State Restoration

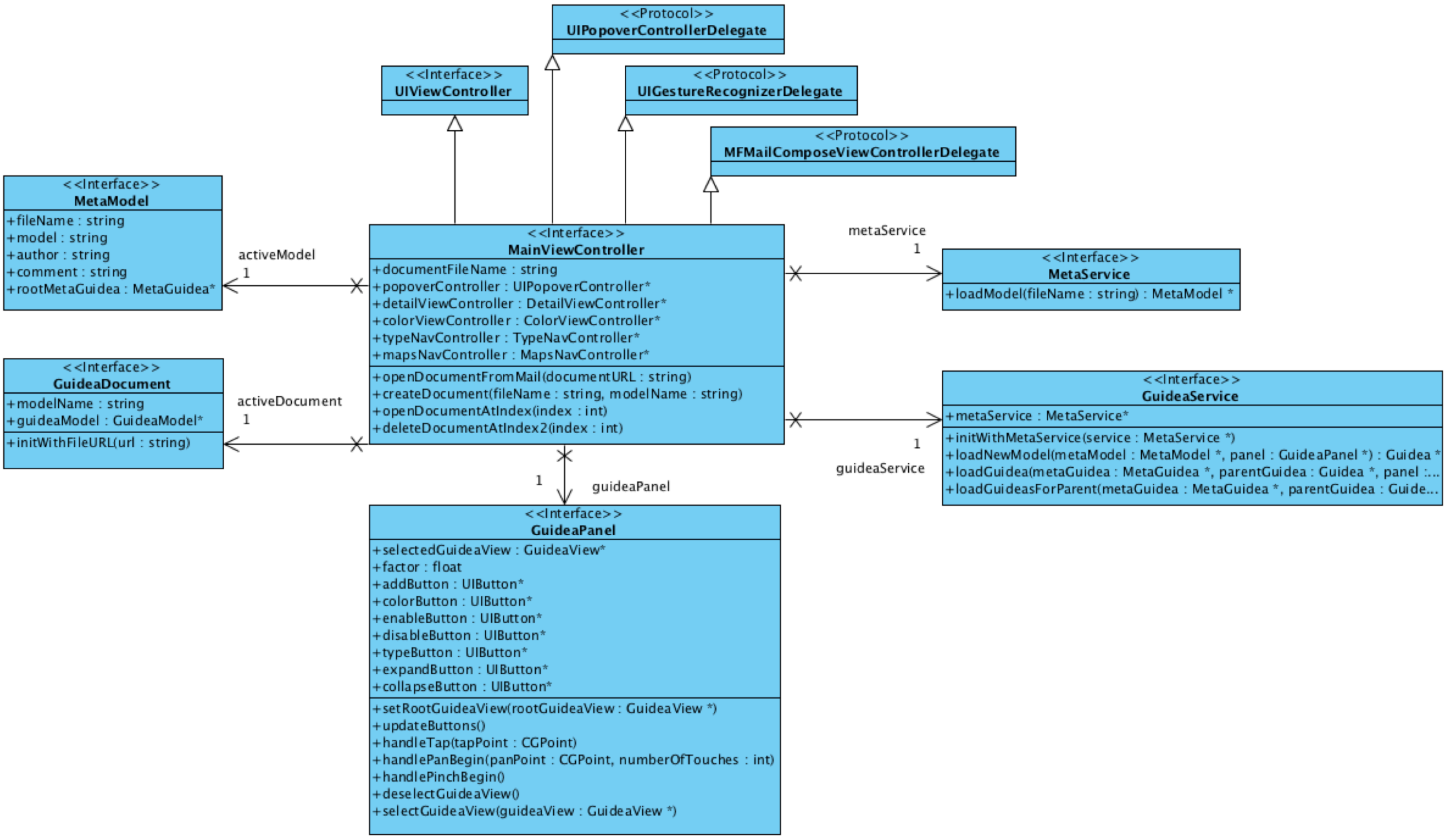
- application:shouldSaveApplicationState:
- application:shouldRestoreApplicationState:
- application:viewControllerWithRestorationIdentifierPath:coder:
- application:willEncodeRestorableStateWithCoder:
- application:didDecodeRestorableStateWithCoder:

Figure-7.8 Fundamental UIApplicationDelegate methods

Methods that are relevant for GuideaMaps application have been overwritten in the **GuideaMapsDelegate** implementation (see snippet-7.1).

7.2.2 The Main Controller

The **`UIViewController`** class provides the fundamental view-management model for iOS apps. The developer will instantiate subclasses of the **`UIViewController`** class based on the specific task each subclass performs. A view controller manages a set of views that make up a portion of the app's user interface. The view controller coordinates its efforts with model objects and other controller objects, including other view controllers, to present a single coherent user interface.



<<Protocol>>
UIPopoverControllerDelegate
<<Interface>>
UIViewController
<<Protocol>>
UIGestureRecognizerDelegate
<<Protocol>>
MFMailComposeViewControllerDelegate
<<Interface>>
MetaModel
+fileName : string
+model : string
+author : string
+comment : string
+rootMetaGuidea : MetaGuidea*
activeModel
1
<<Interface>>
MainViewController
+documentFileName : string
+popoverController : UIPopoverController*
+detailViewController : DetailViewController*
+colorViewController : ColorViewController*
+typeNavController : TypeNavController*
+mapsNavController : MapsNavController*
+openDocumentFromMail(documentURL : string)
+createDocument(fileName : string, modelName : string)
+openDocumentAtIndex(index : int)
+deleteDocumentAtIndex2(index : int)
metaService
1
<<Interface>>
MetaService
+loadModel(fileName : string) : MetaModel *
<<Interface>>
GuideaDocument
+modelName : string
+guideaModel : GuideaModel*
+initWithFileURL(url : string)
activeDocument
1
<<Interface>>
GuideaService
+metaService : MetaService*
+initWithMetaService(service : MetaService *)
+loadNewModel(metaModel : MetaModel *, panel : GuideaPanel *) : Guidea *
+loadGuidea(metaGuidea : MetaGuidea *, parentGuidea : Guidea *, panel :...
+loadGuideasForParent(metaGuidea : MetaGuidea *, parentGuidea : Guide...
1
guideaService
1
guideaPanel
<<Interface>>
GuideaPanel
+selectedGuideaView : GuideaView*
+factor : float
+addButton : UIButton*
+colorButton : UIButton*
+enableButton : UIButton*
+disableButton : UIButton*
+typeButton : UIButton*
+expandButton : UIButton*
+collapseButton : UIButton*
+setRootGuideaView(rootGuideaView : GuideaView *)
+updateButtons()
+handleTap(tapPoint : CGPoint)
+handlePanBegin(panPoint : CGPoint, numberOfTouches : int)
+handlePinchBegin()
+deselectGuideaView()
+selectGuideaView(guideaView : GuideaView *)

Figure-7.9 GuideaMaps MainViewController in UML 2.0

The view controller:

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

View controllers are tightly bound to the views they manage and take part in the responder chain used to handle events. View controllers are descendants of the **`UIResponder`** class and are inserted into the responder chain between the managed root view and its superview, which typically belongs to a different view controller.

If the view controller's view does not handle an event, the view controller has the option of handling the event or it can pass the event to the superview.

The **`MainViewController`** (see figure-7.9) is the central controller of the GuideaMaps application and is responsible for the coordination between the model that is kept in the **`GuideaDocument`** and the visualization in the **`GuideaPanel`**. It has references to the **`MetaService`** and **`GuideaService`** objects and implements a number to delegate methods to handle popovers, gestures and mail support.

View controllers are rarely used in isolation. Instead, the developer uses multiple view controllers, each of which owns a portion of the app's user interface.

GuideaMap includes a set of view controllers to handle parts of the application:

- **`DetailViewController`** : popover controller for setting guidea details
- **`ColorViewController`** : popover controller for specifying the color
- **`TypeNavController`** : navigation controller for type popover
- **`TypeTableController`** : table controller for list of available types
- **`TypeDetailController`** : table controller for type details
- **`MapsNavController`** : navigator controller for saved maps
- **`MapsTableController`** : table controller for list of saved maps
- **`NewMapController`** : form sheet controller for adding new map

7.2.3 Storyboards[23]

Interface Builder is an application that let the developer build his interfaces visually. Built-in objects like buttons, text fields, tab bars, sliders and labels can easily be dragged onto the app's interface and configured by tweaking the corresponding palettes and panels.

The developer can also use Interface Builder to connect targets and actions (what happens when an interface element is acted on by a user, and what object handles the action) as well as manipulate controllers and object bindings.

A storyboard is a visual representation within Interface Builder of the user interface of an iOS application, showing screens of content and the connections between those screens. A storyboard is composed of a sequence of scenes, each of which represents a view controller and its views; scenes are connected by segue objects, which represent a transition between two view controllers.

Xcode provides a visual editor for storyboards, where the developer can lay out and design the user interface of your application by adding views such as buttons,

[23] http://developer.apple.com/library/mac/#documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html

table views, and text views onto scenes. In addition, a storyboard enables the developer to connect a view to its controller object and to manage the transfer of data between view controllers.

Using storyboards is the recommended way to design the user interface of your application because they enable the developer to visualize the appearance and flow of his user interface on one canvas.

The GuideaMaps project includes view controllers that are defined in the Interface Builder storyboard editor for each of the graphical windows and popovers of the application (see figure-7.10).

Changes to the UI can be made in this editor without having to make modifications to the source code.

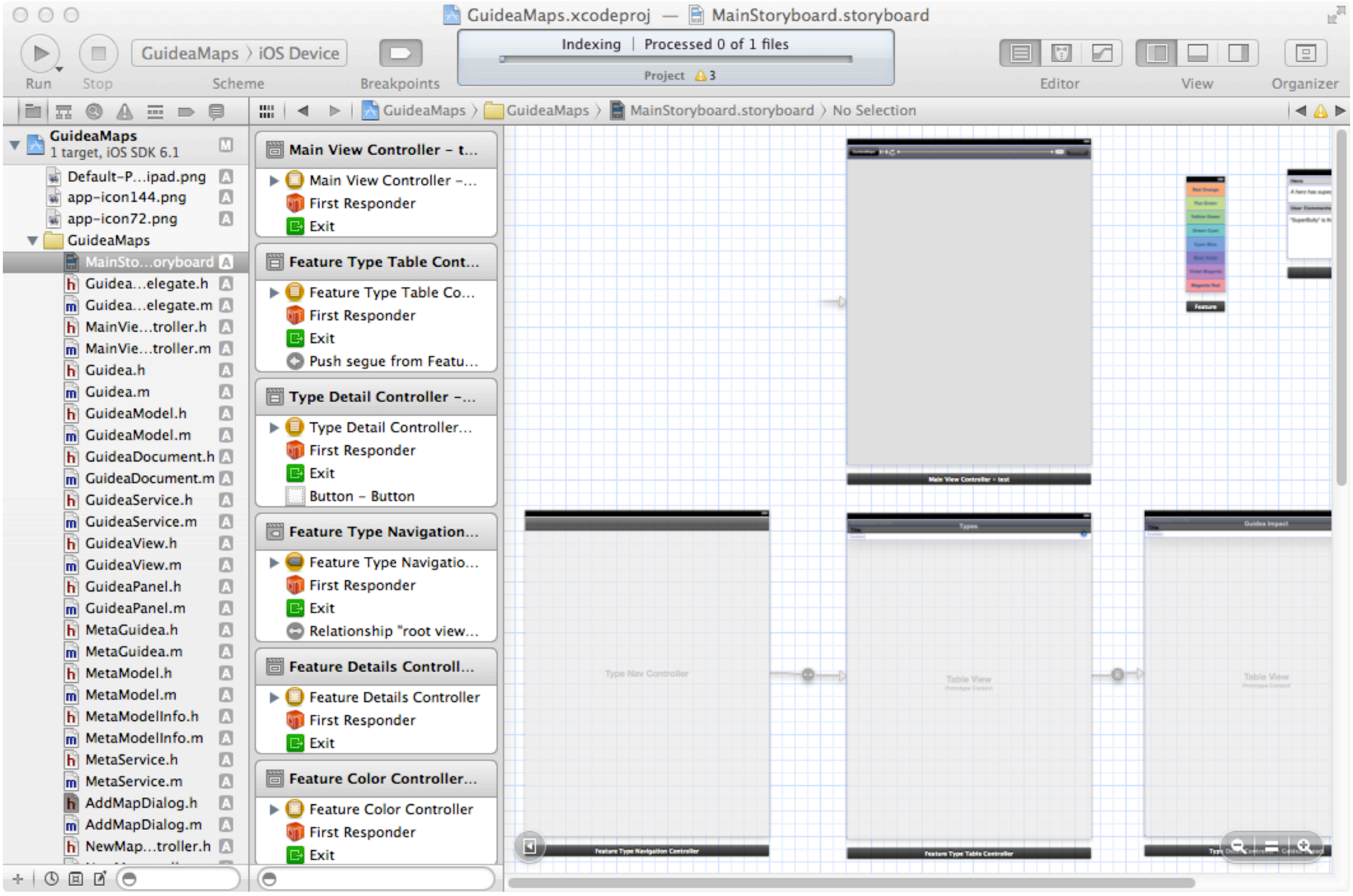


Figure-7.10 GuideaMaps storyboard

For each controller, its view and sub-views are represented using a tree-based structure. This provide a very simple way to navigate to the individual elements to set their specific attributes and properties.

The developer can also specify a “first responder”; the responder object that first receives events. These events include key events, motion events, and action messages, among others. Figure-7.11 show the tree structure for a view and the first responder.

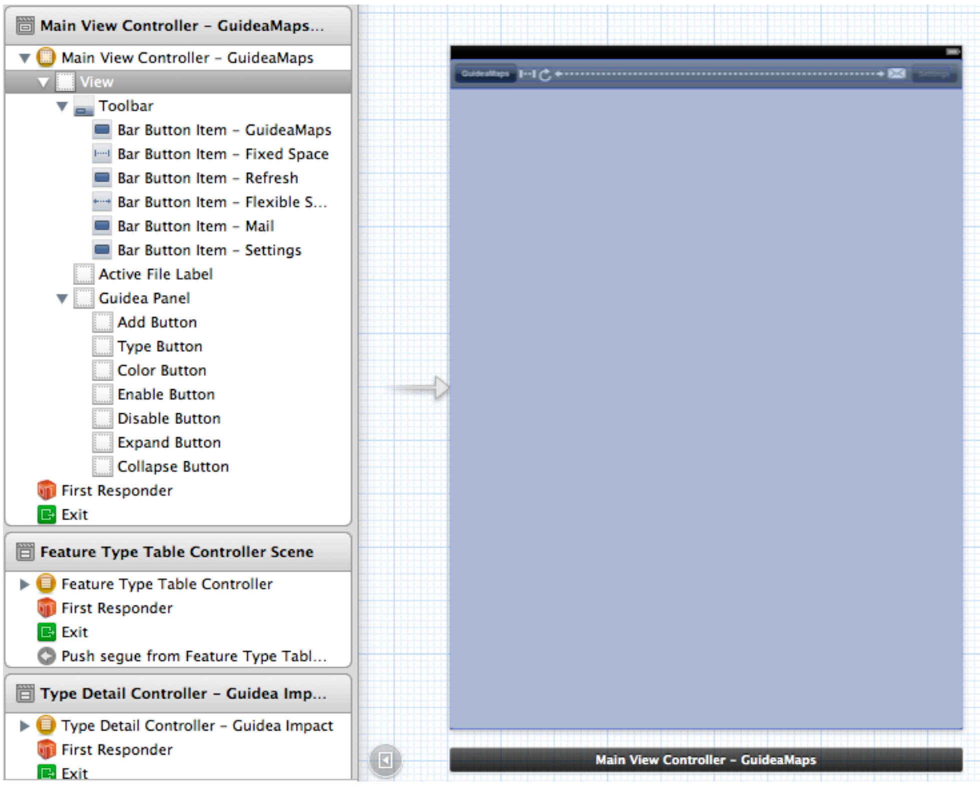
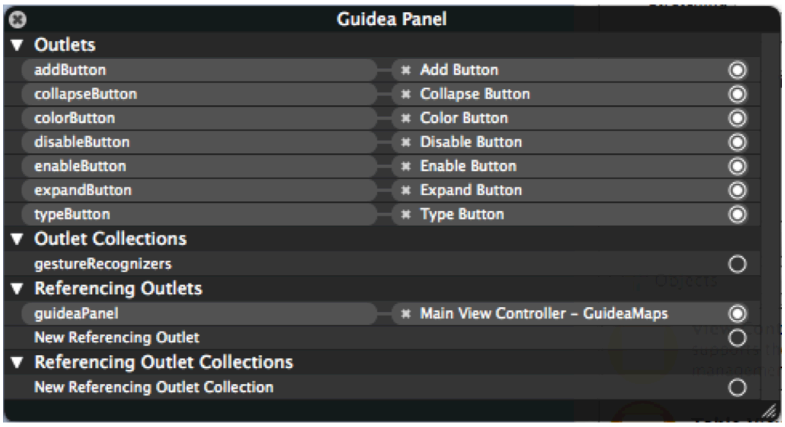

Main View Controller – GuideaMaps...
Main View Controller – GuideaMaps
View
Toolbar
Bar Button Item – GuideaMaps
Bar Button Item – Fixed Space
Bar Button Item – Refresh
Bar Button Item – Flexible S...
Bar Button Item – Mail
Bar Button Item – Settings
Active File Label
Guidea Panel
Add Button
Type Button
Color Button
Enable Button
Disable Button
Expand Button
Collapse Button
First Responder
Exit
Feature Type Table Controller Scene
Feature Type Table Controller
First Responder
Exit
Push segue from Feature Type Tabl...
Type Detail Controller – Guidea Imp...
Type Detail Controller – Guidea Impact
First Responder
Exit
Main View Controller – GuideaMaps

Figure-7.11 Main View Controller view and sub-views

Within the Interface Builder, the developer can define the links between the graphical elements and the source code of the GuideaMaps application; he can define links (“outlets”) to other components (e.g. buttons, text fields, controllers, etc.) without having to hardcode this relationship in the source code (see figure-7.12).



Guidea Panel

▼ Outlets

addButton	Add Button ●
collapseButton	Collapse Button ●
colorButton	Color Button ●
disableButton	Disable Button ●
enableButton	Enable Button ●
expandButton	Expand Button ●
typeButton	Type Button ●

▼ Outlet Collections

gestureRecognizers ○

▼ Referencing Outlets

guideaPanel	Main View Controller – GuideaMaps ●
New Referencing Outlet	○

▼ Referencing Outlet Collections

New Referencing Outlet Collection ○

Figure-7.12 Defining Outlets for the GuideaPanel

This advanced separation of UI and source code is a very efficient way of building GUI applications in iOS.

7.2.4 Services, Guideas, and MetaGuideas

Figure-7.13 shows the core services of the GuideaMaps application; these two services are responsible for maintaining the feature and the application model.

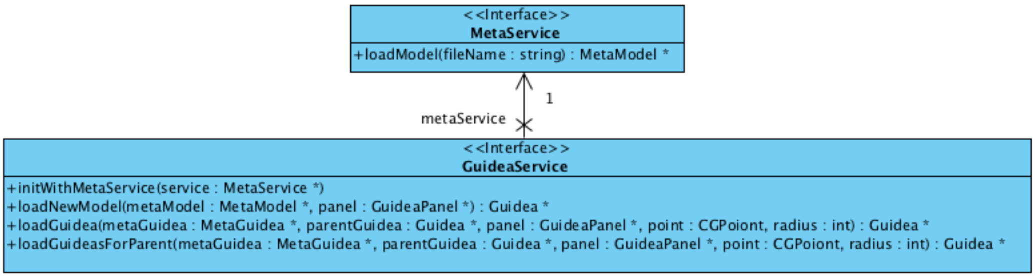

<<Interface>>
MetaService
+loadModel(fileName : string) : MetaModel *
1
metaService
<<Interface>>
GuideaService
+initWithMetaService(service : MetaService *)
+loadNewModel(metaModel : MetaModel *, panel : GuideaPanel *) : Guidea *
+loadGuidea(metaGuidea : MetaGuidea *, parentGuidea : Guidea *, panel : GuideaPanel *, point : CGPoiont, radius : int) : Guidea *
+loadGuideasForParent(metaGuidea : MetaGuidea *, parentGuidea : Guidea *, panel : GuideaPanel *, point : CGPoiont, radius : int) : Guidea *

Figure-7.13 GuideaMaps core services in UML 2.0

The **MetaService** is responsible for loading a given metaModel from an XML file into memory.

The **GuideaService** includes a reference to the MetaService that is provided during the initialization of the service. It is responsible for arranging a new application model (based on a loaded metaModel) and visualizing it in a given **GuideaPanel**. It is also responsible for adding guideas from the metaModel when requested by the user.

The **MetaGuidea** and **Guidea** classes represent the decision and application model

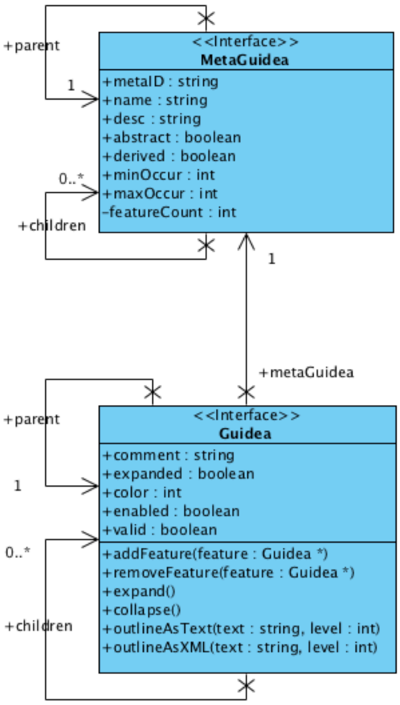


Figure-7.14 MetaGuideas and Guideas in UML 2.0

of the GuideaMaps application (see figure-7.14).

Both classes are instantiated using a dynamic structure; every instance has an optional reference to its parent and a list of children. The **`MetaGuidea`** structure represents the information from decision model and captures information like the name, description, abstract / derived Boolean, cardinality, etc.

The **`Guidea`** structure embodies the application model; it represents the decisions that have been selected by the user including the provided comment and color settings. Screen coordinates are not stored in these objects; a **`GuideaView`** object with x/y view coordinates refers to each **`Guidea`** instance.

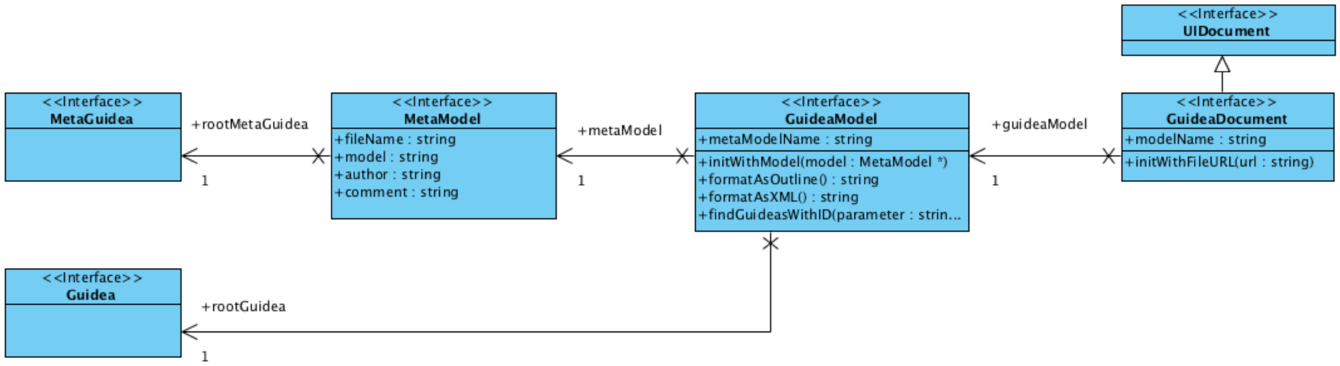


Figure-7.15 GuideaModel and MetaModel in UML 2.0

The **`GuideaDocument`** (see figure-7.15) extends the iOS **`UIDocument`** interface that is responsible for loading and saving a file into the application. The data of the document is captured into a class that implements the **`GuideaModel`** interface. This model represents the application model and has references to the root **`Guidea`** object and the corresponding metaModel. Within this **`MetaModel`**, a reference is kept to the root metaGuidea.

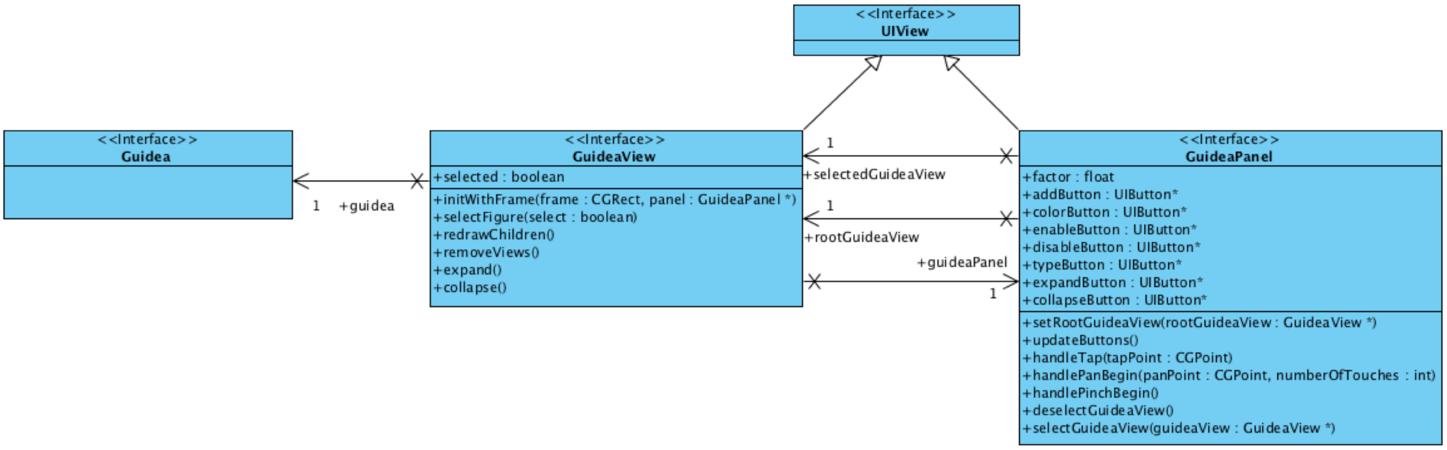

<<Interface>>
UIView
<<Interface>>
Guidea
1 +guidea
<<Interface>>
GuideaView
+selected : boolean
+initWithFrame(frame : CGRect, panel : GuideaPanel *)
+selectFigure(select : boolean)
+redrawChildren()
+removeViews()
+expand()
+collapse()
1
+selectedGuideaView
1
+rootGuideaView
+guideaPanel
1
<<Interface>>
GuideaPanel
+factor : float
+addButton : UIButton*
+colorButton : UIButton*
+enableButton : UIButton*
+disableButton : UIButton*
+typeButton : UIButton*
+expandButton : UIButton*
+collapseButton : UIButton*
+setRootGuideaView(rootGuideaView : GuideaView *)
+updateButtons()
+handleTap(tapPoint : CGPoint)
+handlePanBegin(panPoint : CGPoint, numberOfTouches : int)
+handlePinchBegin()
+deselectGuideaView()
+selectGuideaView(guideaView : GuideaView *)

Figure-7.16 GuideView and GuidePanel in UML 2.0

The **GuideaView** (see figure-7.16) is a visual representation of a **Guidea** within a **GuideaPanel**. The **GuideaView** contains the logic to draw and select or deselect a **Guidea**.

The **GuideaPanel** encapsulates all the **GuideaViews** that make up the active decision model and provide the necessary methods to respond to the user gestures (tap, pan and pinch). It also contains references to the individual action buttons (add, color, type, etc.) and the current zoom factor.

7.2.5 Loading the Models

Both the decision and application model are stored in XML files. To read these files, a parser that is included in the iOS frameworks was used. The NSXMLParser[24] class uses an event-driven style of parsing.

An **NSXMLParser** object sends messages to its delegate; it sends a different message for each type of parsing event. As the parser sequentially encounters each item in an XML file—element, attribute, declaration, entity reference, and so on—it reports it to its delegate (if the delegate implements the associated method), along with any surrounding context.

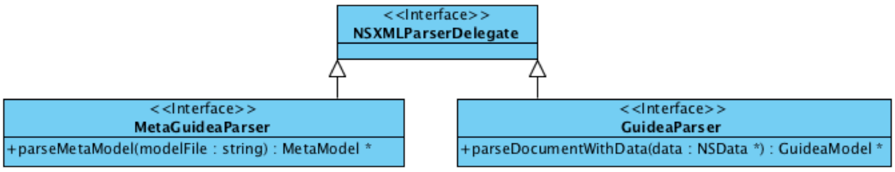

<<Interface>>
NSXMLParserDelegate
<<Interface>>
MetaGuideaParser
+parseMetaModel(modelFile : string) : MetaModel *
<<Interface>>
GuideaParser
+parseDocumentWithData(data : NSData *) : GuideaModel *

Figure-7.17 MetaGuideaParser and GuideaParser in UML 2.0

MetaGuideaParser and **GuideaParser** (see figure-7.17) instantiate a **NSXMLParser** object and implement the required **NSXMLParserDelegate** protocol to capture the elements and attributes from the XML file (see snippet-7.4).

```
@implementation GuideaParser

-(GuideaModel *)parseDocumentWithData:(NSData *)data {
    guideaModel = [[GuideaModel alloc]initWithModel:nil];
    NSXMLParser *xmlparser = [[NSXMLParser alloc] initWithData:data];

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

    // now parse the document
    BOOL ok = [xmlparser parse];
```

[24] http://developer.apple.com/library/ios/#documentation/cocoa/reference/NSXMLParserDelegate_Protocol/Reference/Reference.html

```
}

-(void)parserDidStartDocument:(NSXMLParser *)parser {…}

-(void)parserDidEndDocument:(NSXMLParser *)parser {…}

-(void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName
     namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName
     attributes:(NSDictionary *)attributeDict {…}

-(void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName {…}

- (void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string {…}

-(void)parser:(NSXMLParser *)parser parseErrorOccurred:(NSError *)parseError
{…}

@end
```

Snippet-7-4 GuideaParser and its delegate methods

7.3 Distributing the Software[25]

The iOS Developer Program provides the developer with the ability to distribute his applications to the App Store. He can offer his free or commercial apps to millions of iPad, iPhone, and iPod touch customers around the world.

With Ad Hoc distribution, the developer can share his app with up to 100 iOS devices via email or his server.

To submit an app to the store, the developer uses Xcode features and several web tools available only to members of an Apple Developer Program. Even before he can develop with technologies, such as iCloud and Game Center, he must join the Apple Developer Program for iOS. The developer should join a program even if he distributes applications outside of the Mac App Store so that customers know the application comes from a known source.

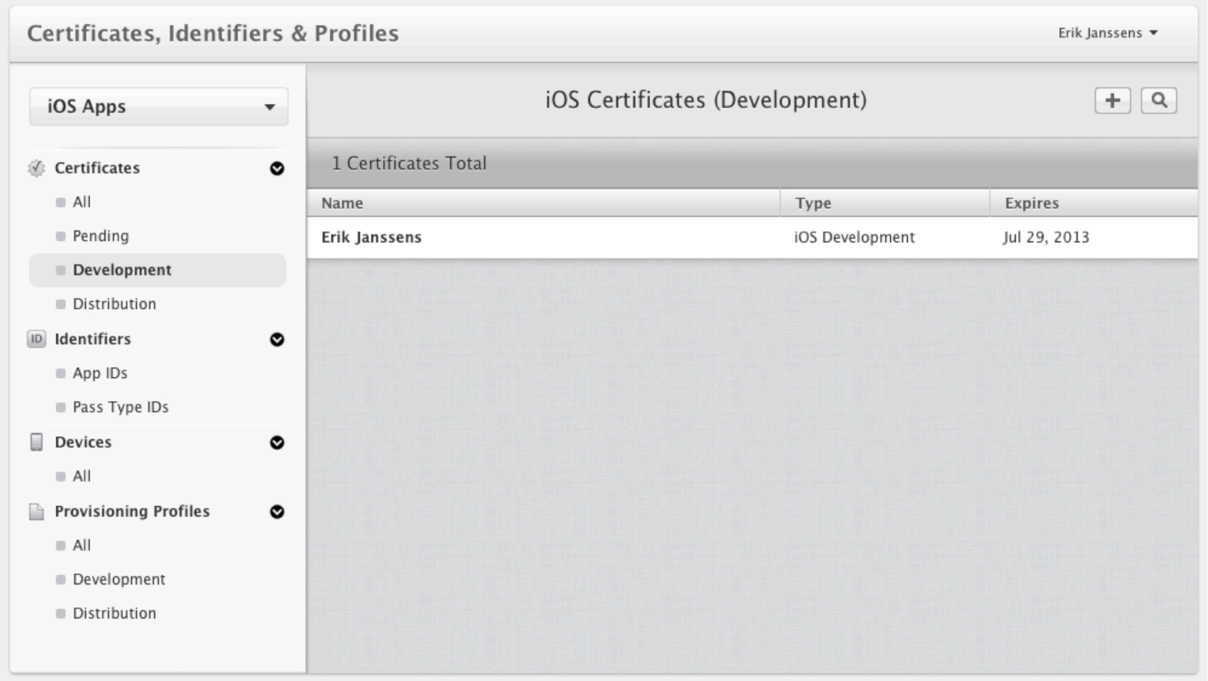
Once the developer has joined a program, he can start using store services from his app, testing his app on devices, providing marketing, sales, and contact information, and submitting versions of the app for approval. The developer iterates the steps of this distribution process, as necessary, until the app is approved and released. Then the developer repeats some of these steps again for each subsequent update.

Use Apple's iOS Provisioning Portal[26] the developer can setup his development and distribution certificates. He can also define the required application IDs, manage the list of development device and the iOS provisioning profiles.

[25] http://developer.apple.com/library/ios/#documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html

A provisioning profile is a collection of digital entities that uniquely ties developers and devices to an authorized iPhone Development Team and enables a device to be used for testing.

A registered developer can login to the portal and manage the necessary artefacts to deploy his app on development devices or Apple's App Store (see figure-7.18).

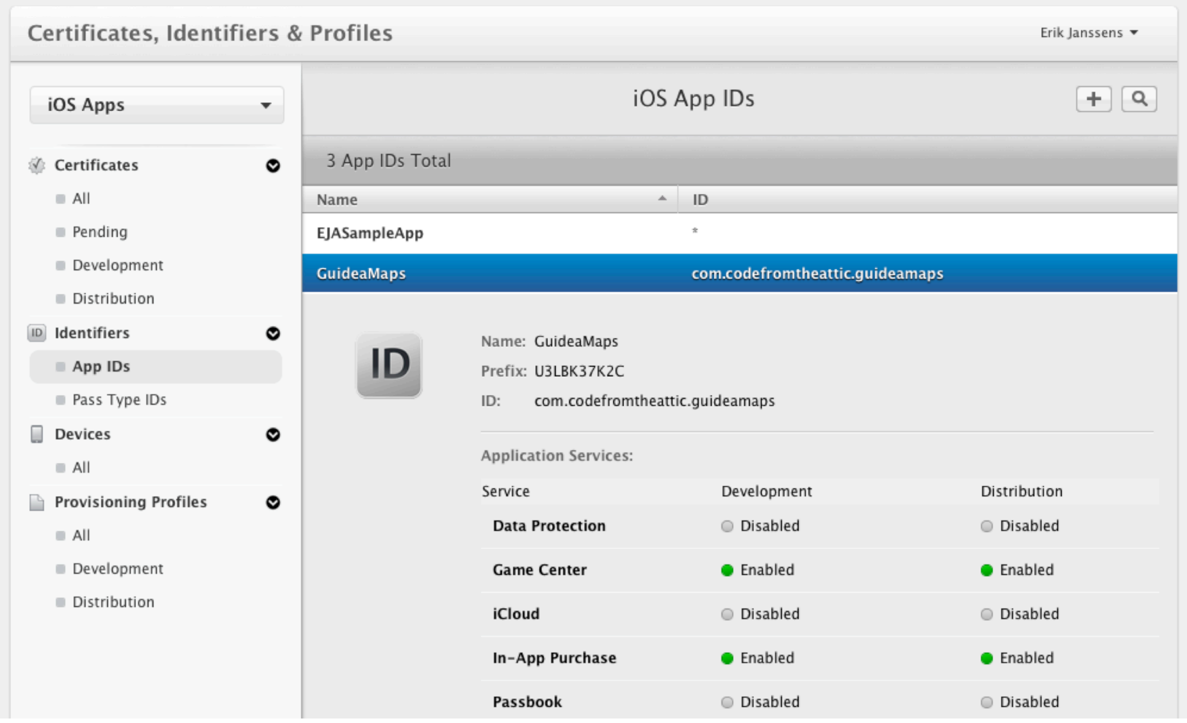


Name	Type	Expires
Erik Janssens	iOS Development	Jul 29, 2013

Figure-7.18 Setting up certificates in the Provisioning Portal

[26] https://developer.apple.com/account/

As shown in figure-7.19, the developer can also define unique App IDs and specify the Application Services that are required within his applications (e.g. iCloud support, in-app purchases).



3 App IDs Total

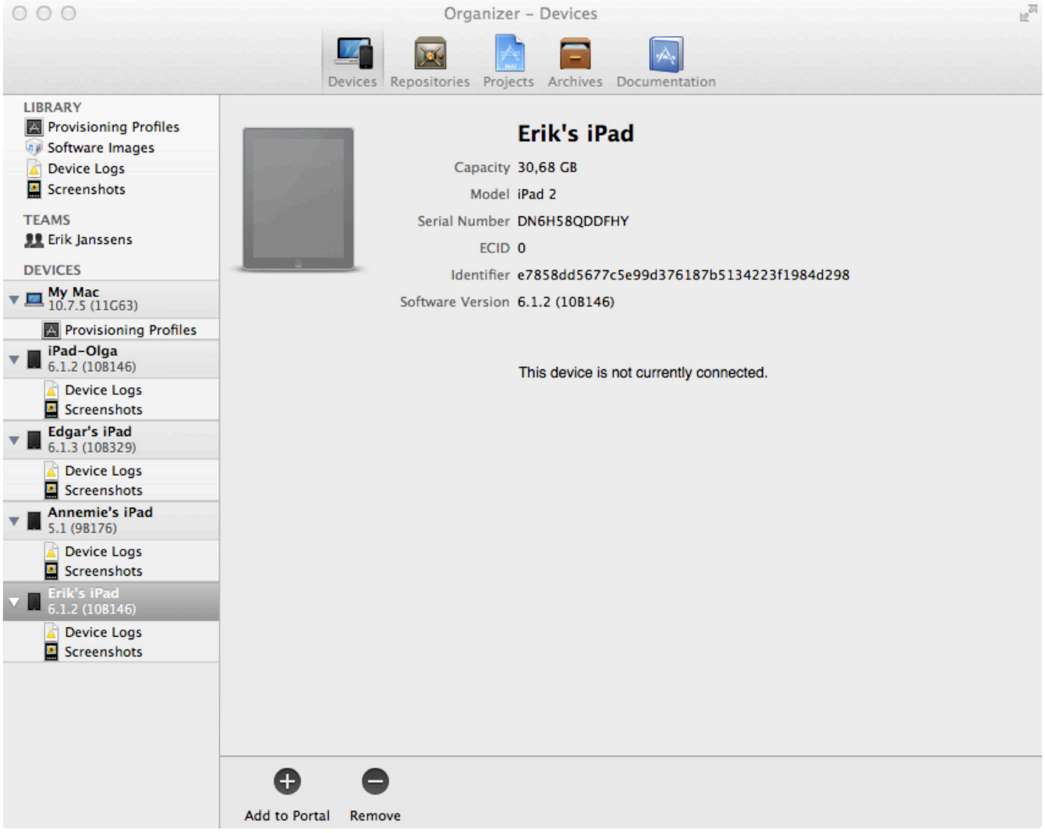
Name	ID
EJASampleApp	*
GuideaMaps	com.codefromtheattic.guideamaps

Name: GuideaMaps
Prefix: U3LBK37K2C
ID: com.codefromtheattic.guideamaps

Application Services:

Service	Development	Distribution
Data Protection	○ Disabled	○ Disabled
Game Center	● Enabled	● Enabled
iCloud	○ Disabled	○ Disabled
In-App Purchase	● Enabled	● Enabled
Passbook	○ Disabled	○ Disabled

Figure-7.19 Defining App IDs



Erik's iPad

Capacity 30,68 GB
Model iPad 2
Serial Number DN6H58QDDFHY
ECID 0
Identifier e7858dd5677c5e99d376187b5134223f1984d298
Software Version 6.1.2 (10B146)

This device is not currently connected.

Figure-7.20 Development devices registered in Xcode

The developer can also configure some of the above information directly from within the Xcode IDE (see figure-7.20).

8 Validation & Future Work

In this chapter presents the validation of the tool (section 8.1) as well as possible future work (section 8.2).

8.1 Validating the GuideaMaps Application

The validation process for this thesis was twofold; first, the GuideaMaps application was validated by a number of potential users of the application. This is described in section 8.1.1. Based on their feedback as well as on our own experiences with using the tool, future improvements could be identified. This is described in section 8.2. Secondly, the tool was also used in a domain different from serious games, i.e. web design. This is described in section 8.1.2.

8.1.1 Pilot Validation

On May 22nd 2013, we conducted a pilot validation with team members of the Friendly ATTAC[27] project. Katrien Van Cleemput and Sara Bastiaensens have been involved in previous "Serious Game definition" meetings of the Friendly ATTAC project and volunteered to assist in this validation session (picture-8.1). The two persons involved in this pilot evaluation had a background in Communication Science and had very little experience with tablets.

Picture-8.1 Pilot Validation

Because, during the previous "Serious Game definition" meetings of the Friendly ATTAC project, most of the requirements for the serious game to be developed were already discussed (although in an ad-hoc way), in this validation session we focused on the following objectives:

[27] http://www.friendlyattac.be

- To validate the usability of the tool.
- To validate the Decision Model that was assembled to capture the requirements of a Serious Game.

It must be noted that given the limitations of this evaluation setup, the usefulness of the tool in the context of requirement elicitation meetings with a large number of people has not been tested.

After a short introduction explaining the goals of the session and a short demonstration of the tool (5 minutes), the first part of the session was started; a preconfigured iPad with the GuideaMaps application and the "Serious Game" Decision Model preinstalled was handed over to the project members.

The participants were asked to enter the available information about their serious game (gathered in the previous "Serious Game definition" meetings of the Friendly ATTAC project) in the tool while we monitored their behavior. The goal of this exercise was to detect inconsistencies and ambiguous UI interactions in the application. It would also allow us to discover missing guideas in the model. Getting started and entering the information took about 40 minutes. This included some discussions between the two participants about what exactly had been decided in previous meetings. The output can be found in Appendix B.

After having entered all their information, we asked the participants for feedback, suggestions for improvement, and the completeness/relevance of the provided Decision Model. This discussion lasted 10 minutes.

We noticed that the participants had little experience with working with iPad applications; typical iOS-related UI interactions (how to dismiss a popover dialog, hide the on-screen keyboard) were sometimes a challenge. This is not specific for the GuideaMaps application, but after a few trials, the participants picked up the iPad interaction approach and used it correctly.

Once the participants were acquainted with the GuideaMaps user interface, they experienced little problems in using the interface except for the fact that one of the participants was constantly mixing the "?" and "+" buttons. When she wanted to select a concrete guidea, she tapped on the "+" button. For the other participant, this was not an issue. A more in-depth usability study might be needed to validate the usability of these icons.

Some issues were raised during the session:

- The first time a user is working with the Decision Model he does not know if a specific guidea is included in the model or not. Therefore, he needs to browse through the model to confirm this. Once the user has been working

with the model, he becomes more familiar with the structure and understands its related guideas.
- When entering information (e.g., "Age Range"), the participants were wondering where they could enter information about the reasons for making a specific choice? For the moment, we opted to provide one big "unstructured" field (user comments) over multiple specific fields due to screen limitations. It should be further investigated whether providing more fields (reason, priority, …) provides added value.

Some good suggestions were identified for potential new features:

- Include an indication (color, bar graph, …) of the completeness of a parent guidea. While walking through the Decision Model's structure and completing information, the user has to remember which parts have been finalized. A graphical representation would avoid that the user is going back to previously visited elements of the graph structure and quickly sees what still needs to be considered.
- The previous suggestion could be enriched by user-selectable labels: e.g. "Completed", "Not Yet Decided", "Not Relevant" to provide status information about the information capturing process.The description of an abstract guidea could be displayed in the list of selectable concrete guideas. This could improve the information that is needed by the user to select the proper guidea from the list.
- If the multiplicity of an abstract guidea is greater than 1, it would be useful to allow the user to select more than one concrete item from the guideas list at once. Some investigation is needed to validate if this is supported by the iOS UI guidelines.
- It would be useful to investigate if it is possible to automatically eliminate options for abstract guideas based on the selection of another guideas (using the excluded dependencies)
- The color of a guidea is always initially set to orange. By including color information in the Decision Model, different colors could be used automatically for sub-models which may be easier for the user to see with which sub-model he is working.

The structure of the Decision Model used, was easily recognized by the participants and after some initial doubts the users felt familiar with it. A number of refinements were proposed with respect to the dependencies. Some minor flaws were detected (e.g. "Punishment" should be optional). Also it was suggested to add a "Hosting" guidea to the model, to capture the requirements about how the game should be hosted after the development. Furthermore, names of different guideas should be clarified in a number of cases (e.g., "What do you mean by technology?")

We also discussed the usefulness of allowing the user to add new guideas while using a GuideaMap. This could solve the problem of "missing" guideas and prevent that the user has to wait until the Decision Model has been updated, on the other hand users may introduce guideas that are already available, resulting in redundant and possibly messy specifications.

A number of bugs were detected (most of them were already known):

- In some situations, the on-screen keyboard is hiding the selected guidea.
- Long names and explanations are not displayed completely (due to space limitations).

The participants were impressed by the functionality to indicate (possible) conflicts ("Cool!"), the green/red icons in the list of selectable guideas ("Handy") and the structured representation of the selected decisions in the email message ("Very useful"). In the second phase, we also discussed the user's first experience with the tool and potential overall improvements.

- The participants didn't enter a lot of information in the "user comments" field. We asked them why. They answered that typing long text messages was not easy on a tablet; adding more information later on, on a PC/Mac was suggested as a solution. This could be a valid remark although given the fact that the users were novice tablet users, this comment might be less relevant (some users write complete books on tablets).
- Collaborative features to allow multiple users to work on the same model was considered as a useful extension.

In general, the participants provided positive feedback regarding the usage of the application and were impressed by the completeness of the "Serious Game" Decision Model. They concluded that the tool could be "very useful in brainstorming meetings to capture, in a structured way, the different decisions and make the necessary progress".

8.1.2 Alternative Domains

The tool is not limited to the requirement elicitation for a serious game. With dedicated Decision Models it can also be used for the requirement elicitation of other types of applications. As an example, a Decision Model was created to use the tool to support a web designer to design a website based on a website genre. An example of a website genre is for instance an e-commerce website (i.e. e-shop). This was done as follows.

A website genre can be associated with a model describing all mandatory and all optional features to be included in a website for that genre, as well as the dependencies between the different features. For this, feature modelling can be used. The resulting feature model can on its turn be considered as a Decision Model for our tool. Using this Decision Model, the tool supports a web designer in selecting all desired features for his website. Figure-8.1 shows the app in use for the specification of an e-commerce website.

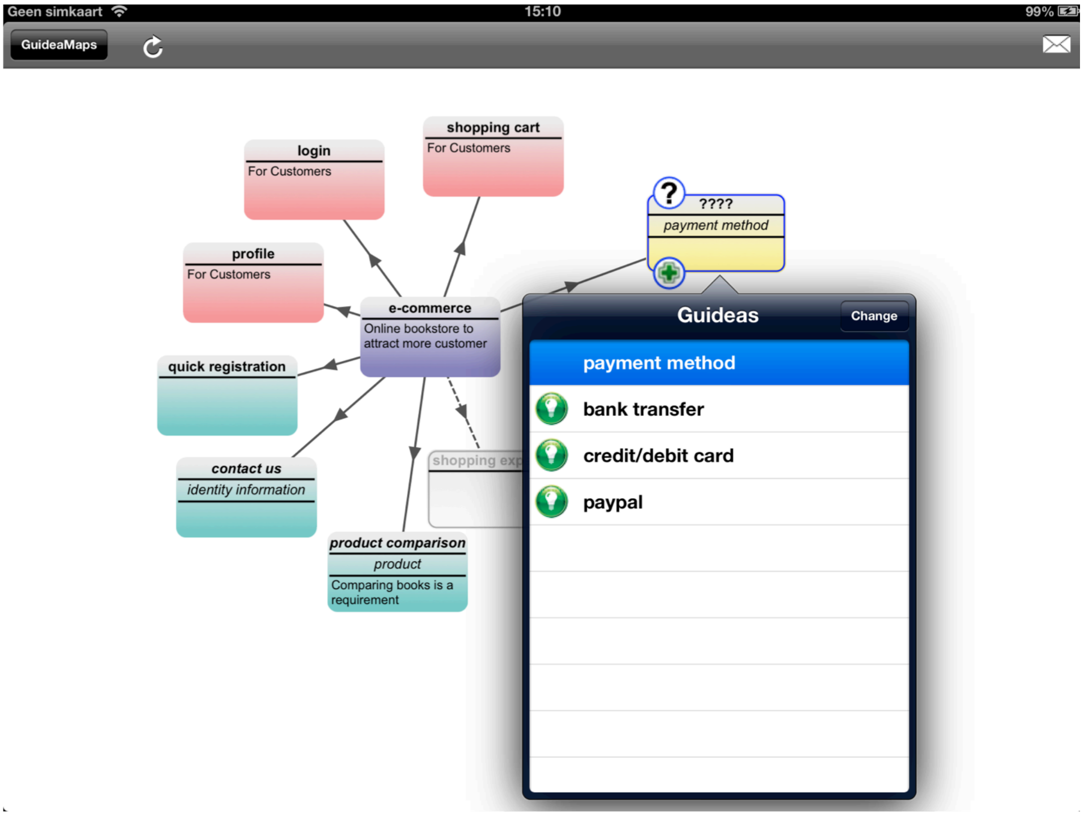


Figure-8.1 Specifications for e-commerce website in GuideaMaps

8.2 Future Work

The GuideaMaps application presented in the document is not a final product; it is merely an advanced prototype that implements the specified requirements, visualizes the proposed Decision Model, and guides the user through the requirement elicitation process.

A number of future features or improvements have been identified during the conceptual, implementation, and validation phases that could be incorporated in the final product.

First of all, several improvements regarding the graphical representation and interaction have been identified:

- Improved graphical representation for the guideas.
- Including default color settings for metaGuideas in the Decision Model.
- Improved and/or alternative layout for the GuideaMaps.
- Improved on-screen keyboard positioning.
- Improved screen positioning algorithms to calculate better x/y position when expanding guideas.
- Visual mechanism to group different options from the same abstract guidea.
- Dealing with visualization of long guidea descriptions, as well as long names for guideas.

Improvements related to additional functionality suggested are:

- Guidea conflict resolution to fix semi-automatically discovered required/exclude clashes.
- The possibilities to create multiple guideas at once from the list when multiplicity is greater than 1.
- Indicators for the progress and labels to indicate the status of the requirement elicitation process in (parent) guideas.
- Completeness analysis of the Application Model created.
- Improved user management (add/delete/rename) for the available Decision Models and Application Models
- Improved sharing of the Decision and Application Models between users (using iCloud or DropBox).
- Collaboration support to allow multiple users to work on a shared Application Model.

Furthermore, future work could also be considered in developing different dedicated Decision Models for different types of serious games. In addition, it is also possible to develop Decision Models for other domains that the domain of serious games as already illustrated in the previous section.

Finally, more user evaluations have to be done. Both to evaluate the tool, as well as to evaluate the Decision Models developed so far. Two types of evaluations can be considered: lab evaluations to focus on the usability aspects and field evaluations to evaluate the tool in practice.

9 Conclusions

In this thesis, we have provided an answer to the research question "How to guide non-technical people (i.e. casual users) in the requirement elicitation process of a serious game using (iPad) tablets?". This is an important question as currently serious games receive a lot of attention but there is little support for the development. In general, people with different background are involved in the development and each of these people have there own priorities. In addition, due to lack of experiences, there is little good practice available on how to come up with a successful serious game. Requirement elicitation is an important phase in software development, and there is no reason why the development of serious games would be an exception. Therefore, as a first step in supporting the development of serious games, we should try to support the requirement elicitation process.

To answer our main research question, we used the Design Science Research Methodology. To define the objectives for a solution, we identified a set of high-level functional and usability requirements that must be fulfilled by potential solutions. These are described in Chapter 2.

We also studied existing iPad tools and related work to investigate if any tools or approaches are already available that could be used or customized for our purpose. This is described in Chapter 3.

Chapter 4 describes the underlying approach taken for our solution. We use a feature modelling approach for documenting the decisions that have to be taken and the options that are possible, i.e. the Decision Model. The use of a separated decision model has the advantage that the tool is generic; different decision models are possible for different types of serious games. Based on a given Decision Model, the user can create an Application Model, which captures and documents the decisions taken and the options selected.

Chapter 5 describes the actual solution developed, called GuideaMaps, an app for the iPad.

In Chapter 6, the solution is demonstrated by means of the application of GuideMaps for the Friendly ATTAC project, which aims to develop a serious game against cyber bullying.

In chapter 7, we elaborate on the implementation of the tool. The Decision Model, as well as the Application Model, is expressed in an XML format. In this way, the models can be easily exchanged and transferred from and to other applications.

Evaluation is discussed in Chapter 8. A pilot evaluation has been set up with members from the Friendly ATTAC projects. The participants of this evaluation were very positive about the tool and its potential. Chapter 8 also discusses possible future improvements and extensions of the tool. Next to improvements from a graphical point of view, and some additional functionality, future work could concentrate on turning the tool into a collaborative one such that people can work together on the same GuideaMap.

Finally, the work has been communicated by means of a scientific paper at an international workshop “Let’s talk about Failures: Why was the Game for Children not a Success?” associated with the ACM SIGCHI Conference on Human Factors in Computing Systems at Paris (May 2013).

10 Bibliography

Agustin, M., Chuang, G., Delgado, A., Ortega, A., Seaver, J., & Buchanan, J. W. (2007). Game sketching. *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts* (pp. 36–43). New York, NY, USA: ACM. doi:10.1145/1306813.1306829

Brandt, E., & Messeter, J. (2004). Facilitating collaboration through design games. *Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices - Volume 1* (pp. 121–131). New York, NY, USA: ACM. doi:10.1145/1011870.1011885

Carro, R. M., Breda, A. M., Castillo, G., & Bajuelos, A. L. (2006). A Methodology for Developing Adaptive Educational-Game Environments. In a. u. l. De Bra, P. Brusilovsky, & R. Conejo (Eds.), *Adaptive Hypermedia and Adaptive Web-Based Systems: Second International Conference, AH 2002 Malaga, Spain, May 29 - 31, 2002. Proceedings* (Vol. 2347, pp. 90–99). Berlin, Heidelberg: Springer Berlin / Heidelberg. doi:10.1007/3-540-47952-x_11

Cechticky, V., Pasetti, A., Rohlik, O., & Schaufelberger, W. (2004). XML-Based Feature Modelling. *Software Reuse: Methods, Techniques, and Tools, Volume 310*(Lecture Notes in Computer Science), 101–114.

De Troyer, O., Janssens, E., & Vandebosch, H. (2012). How to Kick Off the Development Process of a Serious Game□? Retrieved from http://workshops.icts.sbg.ac.at/CHIGames2013/files/deTroyer_CHIWS2013.pdf

Duin, H., Hauge, J. B., & Thoben, K.-D. (2009). An ideation game conception based on the Synectics method. *On the Horizon*, *17*(4), 286–295. doi:10.1108/10748120910998344

Furtado, A. W. B. (2006). *Sharpludus: improving game development experience through software factories and domain-specific languages."*. Universidade Federal De Pernambuco, Recife.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns - Elements of Reusable Object-Oriented Software* (1st ed., p. 396). Addison-Wesley Publishing Company.

Goldenberg, J., Mazursky, D., & Solomon, S. (1999). Channeled Ideation Approach. *Journal of Marketing Research*, *36*(2), 200–210.

Gordon, W. J. J. (1961). *Synectics: The Development of Creative Capacity* (p. 180). Harper & Brothers.

IWT. (2012). Friendly Attac. Retrieved from http://www.friendlyattac.be

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). Feasibility Study Feature-Oriented Domain Analysis (FODA), (November).

Kultima, A., Niemelä, J., Paavilainen, J., & Saarenpää, H. (2008). Designing game idea generation games. *Proceedings of the 2008 Conference on Future Play: Research, Play, Share* (pp. 137–144). New York, NY, USA: ACM. doi:10.1145/1496984.1497007

Nadolski, R. J., Hummel, H. G. K., Van Den Brink, H. J., Hoefakker, R. E., Slootmaker, A., Kurvers, H. J., & Storm, J. (2008). EMERGO: A methodology and toolkit for developing serious games in higher education. *Simul. Gaming*, *39*(3), 338–352. doi:10.1177/1046878108319278

Peffers, K., 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. doi:10.2753/MIS0742-1222240302

Sanders, E. (2006). Scaffolds for Building Everyday Creativity. In J. Frascara (Ed.), *Designing Effective Communications* (pp. 65–77). New York, NY: Allworth Press.

Shih, P. C.-P. (2011). *Brainstorming Beyond the Laboratory : Idea Generation Practices in Software Development Firms*. University of California.

Smith, J. D., & Graham, T. C. N. (2010). Raptor: sketching games with a tabletop computer. *Proceedings of the International Academic Conference on the Future of Game Design and Technology* (pp. 191–198). New York, NY, USA: ACM. doi:10.1145/1920778.1920805

Susi, T., Johannesson, M., & Backlund, P. (2007). Serious Games – An Overview.

Vaajakallio, K., Lee, J.-J., & Mattelmäki, T. (2009). "It has to be a group work!": co-design with children. *Proceedings of the 8th International Conference on Interaction Design and Children* (pp. 246–249). New York, NY, USA: ACM. doi:10.1145/1551788.1551843

Vandebosch, H., & Van Cleemput, K. (2008). Defining cyberbullying: a qualitative research into the perceptions of youngsters. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, *11*(4), 499–503. doi:10.1089/cpb.2007.0042

Walrave, M., Demoulin, M., Heirman, W., & Van der Perre, A. (2009). *CYBERPESTEN : Pesten in Bits & Bytes*. Observatorium van de Rechten op het Internet.

Zaid, L. A., Kleinermann, F., & De Troyer, O. (2005). Feature Assembly Modelling, A New Technique for Modelling Variable Software. In J. Shishkov, M. Cordeiro, & B. Virvou (Eds.), *5th International Conference on Software and Data Technologies Proceedings* (p. Volume: 1, pp. pp: 29 – 35). Athens, Greece (2010): SciTePress, ISBN 978-989-8425-22-5.

Appendix A – "Serious Games" Decision Model

```
<?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 - version 1.0</model>
   <author> Olga De Troyer</author>
   <comment>Guidedea template for a serious game</comment>
       <feature id="root">
     <name>My Serious Game</name>
     <description> Start here</description>
     <feature id="uaspects">
        <name>User Aspects</name>
            <description>The game should match the target users/players.
Describe them using the given Guideas</description>
            <feature id="age">
                <name>Age Range</name>
                <description>The age of the player may influence different
aspects of the game. Specify the age range here.</description>
        </feature>
        <feature id="gender" abstract="TRUE" optional="TRUE">
                <name>Gender</name>
                <description>Use this if the game is for only one gender. In
general, boys and girls have different preferences</description>
                <cardinality min="0" max="1"/>
                <feature id="boys" extends="TRUE">
                    <name>Boys/Males</name>
                </feature>
                <feature id="girls" extends="TRUE">
                    <name>Girls/Females</name>
                </feature>
        </feature>
        <feature id="pltype" optional="TRUE">
          <name>Player Type </name>
          <description>Use this if you know that the players belong to a
particular (or several) player type(s)</description>
          <feature id="bymotivation" abstract="TRUE" optional="TRUE">
             <name>Type by Motivation</name>
             <description>This is a classification given by Oblinger
(2004)</description>
             <cardinality min="1" max="4"/>
             <feature id="timekillers" extends="TRUE">
                <name>Time Killers</name>
                <description>Shallow players wanting to exprience quick
rewards</description>
             </feature>
             <feature id="funseekers" extends="TRUE">
                <name>Fun Seekers</name>
                <description>Play for immediate gratification. They will
weigh playing the game against another (or another form of
entertainment)</description>
             </feature>
             <feature id="waanbes" extends="TRUE">
                <name>Wanna-be's</name>
                <description>Driven by a desire to belong to the gaming
community</description>
             </feature>
             <feature id="committed" extends="TRUE">
                <name>Committed</name>
                <description>Driven by deep challenges; have a high
tolerance for frustation in pursuit their objective; are self-
motivated</description>
             </feature>
           </feature>
           <feature id="bypersonality" abstract="TRUE" optional="TRUE">
             <name>Type by Personality</name>
             <description>This is a classification given by Bartle
(1996)</description>
             <cardinality min="1" max="4"/>
             <feature id="explorers" extends="TRUE">
                <name>Explorers</name>
                <description>Prefer discovering areas, creating maps, and
learning about hidden places</description>
             </feature>
             <feature id="achievers" extends="TRUE">
                <name>Achievers</name>
                <description>Prefer to gain "points," levels, equipment and
other concrete measurements in a game</description>
             </feature>
             <feature id="socializers" extends="TRUE">
                <name>Socializers</name>
                <description>Choose to play a game for the social aspect;
they like interacting with other players or computer-controlled characters
with personality</description>
             </feature>
             <feature id="killers" extends="TRUE">
                <name>Killers</name>
                <description>Like competition with other players, and prefer
fighting computer-controlled opponents</description>
             </feature>
           </feature>
        </feature>
        <feature id="perschar" optional="TRUE">
           <name>Personality Charactersitics</name>
           <description>Document any characteristics of the players that
could be relevant</description>
        </feature>
        <feature id="competences" optional="TRUE">
           <name>Competences</name>
           <description>Document any competences of the players that could
be relevant</description>
        </feature>
        <feature id="lstyle" optional="TRUE">
           <name>Learning Style</name>
           <description>Use this if you know that the users have( a)
particular (preferred) learning style(s)</description>
           <feature id="vark" abstract="TRUE" optional="TRUE">
             <name>VARK Learning Styles</name>
             <description>Classification of Fleming</description>
             <cardinality min="0" max="4"/>
             <feature id="visual" extends="TRUE">
                <name>Visual Learners</name>
                <description>Learn best by looking at visualizations and
have a preference for seeing (pictures, diagrams, etc.)</description>
             </feature>
             <feature id="auditory" extends="TRUE">
                <name>Auditory Learners</name>
                <description>Learn best through listening (lectures,
discussions, tapes, etc.)</description>
             </feature>
             <feature id="kinesthetic" extends="TRUE">
                <name>Kinesthetic Learners</name>
                <description>Learn best via experience, moving, touching,
and doing</description>
```

```
             </feature>
             <feature id="readwrite" extends="TRUE">
                <name>Read/Write Learners</name>
                <description>Learn best by reading and writing the
material</description>
             </feature>
          </feature>
        </feature>
        </feature>
        <feature id="contextofuse">
            <name>Context of Use</name>
        <description>It is important to know in which context(s) the serious
game will be used</description>
        <feature id="platform" abstract="TRUE">
           <name>Platform</name>
           <description>On which platform(s) will be serious game be
available</description>
           <cardinality min="1" max="4"/>
           <feature id="pc" extends="TRUE">
              <name>PC</name>
           </feature>
           <feature id="mac" extends="TRUE">
              <name>Mac</name>
           </feature>
           <feature id="smartphone" extends="TRUE">
              <name>Smartphone</name>
           </feature>
           <feature id="tablet" extends="TRUE">
              <name>Tablet</name>
            </feature>
        </feature>
        <feature id="place" abstract="TRUE">
           <name>Place</name>
           <description>Where will the serious game be used</description>
           <cardinality min="1" max="3"/>
           <feature id="home" extends="TRUE">
              <name>At home</name>
           </feature>
           <feature id="school" extends="TRUE">
              <name>At School/Institute or In Organization</name>
           </feature>
           <feature id="At the workplace" extends="TRUE">
              <name>Workplace</name>
            </feature>
        </feature>
        <feature id="availability" abstract="TRUE">
           <name>Availability</name>
           <description>How will the access to the serious game be
organized</description>
           <cardinality min="1" max="1"/>
           <feature id="open" extends="TRUE">
              <name>Open</name>
              <description>In principle everybody can buy/access the serious
game</description>
           </feature>
           <feature id="restricted" extends="TRUE">
              <name>Restricted</name>
              <description>The serious game will only beavailable for
certain groups</description>
           </feature>
        </feature>
       </feature>
        <feature id="paspects">
            <name>Pedag. Aspects</name>
            <description>Describe the pedagogical aspects using the given
Guideas</description>
        <feature id="didgoal" abstract="TRUE">
           <name>Didactical goal</name>
           <description>Describe the didactical goal of the serious game and
try to assign one or more of the given types</description>
           <cardinality min="1" max="7"/>
           <feature id="achange" extends="TRUE">
             <name>Attitude Change</name>
             <description>To change the attitude of the player towards a
certain problem/person/ issue</description>
           </feature>
           <feature id="bchange" extends="TRUE">
             <name>Behavioral Change</name>
             <description>To change the behavior of the player with respect
to a certain problem/person/ issue</description>
           </feature>
           <feature id="awareness" extends="TRUE">
             <name>Awareness</name>
             <description>To raise awareness about a certain problem or
issue</description>
           </feature>
           <feature id="kacquisition" extends="TRUE">
             <name>Knowledge Acquisition</name>
             <description>To increase the knowledge of the player about a
certain problem/issue</description>
           </feature>
           <feature id="pskills" extends="TRUE">
             <name>Practicing Skills</name>
             <description>To let the player practice some skills acquired
in the game or elsewhere</description>
           </feature>
           <feature id="sproblemsolving" extends="TRUE">
             <name>Social Problem Solving</name>
             <description>To learn how to solve a (social)
problem(s)</description>
            </feature>
                <feature id="cproblemsolving" extends="TRUE">
             <name>Cognitive Problem Solving</name>
             <description>To learn how to solve (a) cognitive
problem(s)</description>
            </feature>
        </feature>
        <feature id="didapproch" abstract="TRUE">
           <name>Didactical Approach</name>
           <description>Specify the didactical approach(es) that will be
followed</description>
           <cardinality min="1" max="11"/>
           <feature id="practice" extends="TRUE">
             <name>Practice and Feedback</name>
             <description>Rehearsing a behavior over and over, or engaging
in an activity again and again, for the purpose of improving or mastering
it</description>
           </feature>
           <feature id="learnbydo" extends="TRUE">
             <name>Learning by Doing</name>
             <description>Learn how to do something by doing
it</description>
           </feature>
           <feature id="trial" extends="TRUE">
             <name>Trial and Error</name>
             <description>Characterized by repeated, varied attempts which
are continued until success or until one stops trying. Used for problem
solving</description>
           </feature>
```

```
          <feature id="probbased" extends="TRUE">
             <name>Problem-based Learning</name>
             <description>Learn by performing steps to solve a
problem</description>
          </feature>
          <feature id="casebased" extends="TRUE">
             <name>Cased-based Learning</name>
             <description>Learn by addressing a real world
problem/case</description>
          </feature>
          <feature id="taskbased" extends="TRUE">
             <name>Task-based Learning</name>
            <description>A (daily) task is used for embedding the learning.
E.g., used for language acquisition</description>
            </feature>
                <feature id="questionbased" extends="TRUE">
             <name>Question-based Learning</name>
             <description>Questions are used as the driving force for
acquiring knowledge</description>
          </feature>
          <feature id="collab" extends="TRUE">
             <name>Collaborative Learning</name>
             <description>Students at various levels work together towards
a common goal; the students are responsible for one another
learning</description>
          </feature>
          <feature id="cooper" extends="TRUE">
             <name>Cooperative Learning</name>
             <description>The learners are working in groups to accomplish
the elarning goals</description>
            </feature>
                <feature id="discovery" extends="TRUE">
             <name>Discovery-based Learning</name>
             <description>Uses the child's natural curiosity in order to
facilitate learning; it believes that it is best for learners to discover
facts and relationships for themselves</description>
          </feature>
          <feature id="roleplay" extends="TRUE">
             <name>Roleplay Simulation</name>
             <description>Learners take on the roles of specific characters
or organizations in a contrived setting</description>
            </feature>
        </feature>
        <feature id="pcontext" optional="TRUE">
          <name>Pedagogical Context</name>
          <description>Use this to specify if (and how) the serious game
will be part of a pedagogical package</description>
        </feature>
     </feature>
        <feature id="gaspects">
            <name>Game Aspects</name>
            <description>Describe the different aspects of the game using
the given Guideas</description>
          <feature id="ggenre" abstract="TRUE">
             <name>Game Genre</name>
             <description>What genre(s) of game is (are) targeted. Note
that genres are not strictly defined and may overlap</description>
              <cardinality min="1" max="9"/>
              <feature id="roleplaying" extends="TRUE">
                <name>Role Playing Game</name>
                <description>Players assume the roles of characters in a
fictional setting</description>
              </feature>
                    <feature id="adventure" extends="TRUE">
                <name>Adventure Game</name>
                <description>Game without reflex challenges or action;
normally provide challenges to the player to be solve by interacting with
people or the environment, most often in a non-confrontational
way</description>
             </feature>
             <feature id="detective" extends="TRUE">
                <name>Detective Game</name>
                <description>Game in which some mystery must be
solved</description>
              </feature>
             <feature id="strategy" extends="TRUE">
                <name>Strategy Game</name>
                <description>Game in which decision-making skills have a
high significance in determining the outcome</description>
             </feature>
             <feature id="concentration" extends="TRUE">
                <name>Concentration Game</name>
                <description>Games that highly depends on
concentration</description>
             </feature>
             <feature id="sport" extends="TRUE">
                <name>Sport Game</name>
                <description>Game that simulates the practice of a
sport</description>
             </feature>
             <feature id="simulation" extends="TRUE">
                <name>Simulation Game</name>
                <description>Game that simulate aspects of a real or
fictional reality (such as management, life, flying, etc)</description>
             </feature>
             <feature id="action" extends="TRUE">
                <name>Action Game</name>
                <description>Game that emphasizes physical challenges,
including hand-eye coordination and reaction-time, such a shooter games and
fighting game</description>
             </feature>
             <feature id="puzzle" extends="TRUE">
                <name>Puzzle Game</name>
                <description>Game that emphasizes puzzle solving including
logic, strategy, pattern recognition, sequence solving, and word
completion</description>
           </feature>
        </feature>
        <feature id="gformula" abstract="TRUE">
           <name>Game Formula</name>
           <description>Indicate whether the game will be composed of mini
games or one big game</description>
           <cardinality min="1" max="1"/>
           <feature id="minigames" extends="TRUE">
             <name>Mini Games</name>
             <description>Use this you will be using independent mini
games</description>
           </feature>
                <feature id="singlegame" extends="TRUE">
             <name>Single Game</name>
             <description>Use this if the game is a single (maybe large)
game</description>
            </feature>
        </feature>
        <feature id="players" abstract="TRUE">
           <name>Single/Multiple Player</name>
           <description>Indicate whether the serious game is for one player
or for multiple players</description>
           <cardinality min="1" max="1"/>
               <feature id="singleplayer" extends="TRUE">
```

```
                <name>Single Player Game</name>
                <description>Game played by an individual</description>
              </feature>
                    <feature id="multiplayer" extends="TRUE">
                <name>Multiplayer Game</name>
                <description>Game played by several players</description>
                <feature id="massivelyplayer" optional="TRUE">
                    <name>Massively Multiplayer Game</name>
                   <description>A very large number of players interact
with one another within a virtual game world</description>
                </feature>
              </feature>
        </feature>
        <feature id="gconcept">
           <name>Game Concept</name>
           <description>Explain the main concept/idea of the
game</description>
        </feature>
        <feature id="glength">
           <name>Game Length</name>
           <description>Indicate how long it should approximately take to
play the game</description>
        </feature>
        <feature id="gelements">
           <name>Game Elements</name>
            <description>This part is to indicate what kind of elements you
want to have in the game</description>
            <feature id="dialogs">
              <name>Dialogs</name>
             <description>How will the communication between player(s) and
game be supported</description>
             <feature id="inputplayer" abstract="TRUE">
                <name>Player to Game</name>
                <description>How will the player communicate with the
game</description>
                 <cardinality min="1" max="2"/>
                  <feature id="inputtextual" extends="TRUE">
                     <name>Textual</name>
                  <description>Player is using text</description>
                </feature>
                 <feature id="inputspeech" extends="TRUE">
                     <name>Using Speech</name>
                  <description>Player is using speech</description>
                  <requires>microphone</requires>
                </feature>
             </feature>
             <feature id="outputgame" abstract="TRUE">
                <name>Game to Player</name>
                <description>How will the game communicate to the
player</description>
                 <cardinality min="1" max="2"/>
                  <feature id="outputtextual" extends="TRUE">
                     <name>Textual</name>
                  <description>using text (e.g., text
ballons)</description>
                </feature>
                 <feature id="outputspeech" extends="TRUE">
                     <name>Using Speech</name>
                  <description>Game is using speech</description>
                  <requires>speakers</requires>
                </feature>
             </feature>
                <feature id="playerplayer" abstract="TRUE" optional="TRUE">
                <name>Player to Player</name>
                <description>How will the player communicate with another
player</description>
                <cardinality min="0" max="2"/>
                   <feature id="playertextual" extends="TRUE">
                     <name>Textual</name>
                   <description>Player is using text</description>
                    <requires>multiplayer</requires>
                <feature>
                 <feature id="playerspeech" extends="TRUE">
                     <name>Using Speech</name>
                   <description>Player is using speech</description>
                   <requires>microphone</requires>
                   <requires>multiplayer</requires>
                </feature>
             </feature>
            </feature>
                    <feature id="reward" abstract="TRUE">
              <name>Reward System</name>
             <description>Indicate the kind of reward system you want to
use</description>
             <cardinality min="1" max="4"/>
              <feature id="currency" extends="TRUE">
                <name>In-game Currency</name>
                <description>Player earns money and uses this to buy
things</description>
              </feature>
                    <feature id="extras" extends="TRUE">
                <name>Extras</name>
                <description>Rewards are in the form of extra elements, new
abilities </description>
              </feature>
              <feature id="points" extends="TRUE">
                <name>Points</name>
                <description>Player earns points</description>
             </feature>
             <feature id="progress" extends="TRUE">
                <name>Progress in the Game</name>
                <description>Rewards is by allowing the player to progress
towards the goal</description>
             </feature>
          </feature>
          <feature id="feedback" abstract="TRUE" optional="TRUE">
              <name>Feedback System</name>
             <description>Use this if you want to provide feedback to the
player during the game</description>
             <cardinality min="1" max="4"/>
              <feature id="indicators" extends="TRUE">
                <name>Progress Indicators</name>
              </feature>
                    <feature id="sound" extends="TRUE">
                <name>Sound</name>
              </feature>
              <feature id="enabling" extends="TRUE">
                <name>Enabling/Disabling Actions or Interaction</name>
             </feature>
             <feature id="advices" extends="TRUE">
                <name>Advices or Guidance</name>
             </feature>
          </feature>
          <feature id="punishment" abstract="TRUE" optional="TRUE">
              <name>Punishment System</name>
             <description>Use this if you want to "punish" the player for
incorrect action/answer</description>
             <cardinality min="1" max="4"/>
              <feature id="pointloss" extends="TRUE">
```

```
               <name>Loss of Points</name>
               <requires>points</requires>
              </feature>
                    <feature id="extraloss" extends="TRUE">
               <name>Removal of Extras</name>
               <requires>extras</requires>
              </feature>
              <feature id="setback" extends="TRUE">
               <name>Setback</name>
               <description>Player is set back to a previous point in the
game</description>
                        <requires>progress</requires>
             </feature>
             <feature id="gameover" extends="TRUE">
                <name>Game Over</name>
             </feature>
          </feature>
          <feature id="reflection" abstract="TRUE" optional="TRUE">
              <name>Reflection System</name>
             <description>Use this if you want to build in a kind of
reflection for the player</description>
             <cardinality min="1" max="2"/>
              <feature id="compare" extends="TRUE">
               <name>Compare against Normative Value</name>
               <description>Compare Score/Performance against normative
value</description>
              </feature>
                    <feature id="selfreflection" extends="TRUE">
               <name>Self Reflection</name>
               <description>Allow the player to do some kind of self-
reflection</description>
              </feature>
          </feature>
          <feature id="buddy" optional="TRUE">
             <name>Buddy</name>
             <description>Use this if you want to give the player a buddy
(helper)</description>
             <feature id="buddyrole" abstract="TRUE">
               <name>Buddy' Role</name>
               <description>Indicate the role of the buddy</description>
               <cardinality min="1" max="3"/>
                  <feature id="relationship" extends="TRUE">
                  <name>Build Relationship</name>
                  <description>Buddy attempts to build a relationship with
the player</description>
                  <requires>companion</requires>
                  </feature>
                        <feature id="giveadvice" extends="TRUE">
                  <name>Give Advice</name>
                  <description>Buddy will give advice</description>
                  </feature>
                        <feature id="emotional" extends="TRUE">
                  <name>Give Emotional Support</name>
                  <description>Buddy will support the player
emotionally</description>
                  </feature>
             </feature>
             <feature id="buddytype" abstract="TRUE">
               <name>Buddy Type</name>
               <description>Indicate what kind of buddy to
use</description>
               <cardinality min="1" max="1"/>
                  <feature id="rolemodel" extends="TRUE">
                  <name>Role Model</name>
                  <description>The buddy is a role model</description>
                  </feature>
                        <feature id="pedagent" extends="TRUE">
                  <name>Pedagogical Agent</name>
                  <description>The buddy is more like a
teacher/coach</description>
                  <requires>giveadvice</requires>
                  </feature>
                        <feature id="companion" extends="TRUE">
                  <name>Virtual Companion</name>
                  <description>The buddy is like a friend</description>
                  </feature>
             </feature>
          </feature>
          <feature id="avatar" optional="TRUE">
             <name>Avatar</name>
             <description>Use this if you want to represent your player(s)
by avatar(s)</description>
             <feature id="avatartype" abstract="TRUE">
                <name>Type of Avatar</name>
                <description>Indicate the type of avatar</description>
                <cardinality min="1" max="3"/>
                  <feature id="ideal" extends="TRUE">
                  <name>Ideal Form</name>
                  <description>The character the player has always wanted
to be (e.g., Princess)</description>
                  </feature>
                        <feature id="blankslave" extends="TRUE">
                  <name>Blank Slave</name>
                  <description>Iconic character; the less detail given, the
more opportunity the player has to project himself into that
character</description>
                  </feature>
                        <feature id="iconicchar" extends="TRUE">
                  <name>Iconic Character</name>
                  <description>Will usually be related to the
story/background of the game</description>
                </feature>
             </feature>
             <feature id="avatarchar" abstract="TRUE" optional="TRUE">
                <name>Avatar's Characteristics</name>
                <description>Use this if the avatar needs to have specific
characteristics</description>
                <cardinality min="0" max="4"/>
                  <feature id="simage" extends="TRUE">
                  <name>Similar Age</name>
                  <description>Age should be similar to age of the
player</description>
                  </feature>
                        <feature id="simgender" extends="TRUE">
                  <name>Similar Gender</name>
                  <description>Gender should be similar to gender of the
player</description>
                </feature>
                        <feature id="transcending" extends="TRUE">
                  <name>Transcending</name>
                  <description>The avatar should be more "powerful" than
player in reality</description>
                 </feature>
                        <feature id="similar" extends="TRUE">
                  <name>Similar</name>
                  <description>The avatar should be similar to the
player</description>
                  </feature>
             </feature>
          </feature>
```

```
        </feature>
     </feature>
        <feature id="iaspects">
            <name>Implem. Aspects</name>
        <description>Describe the implementation aspects using the given
Guideas</description>
        <feature id="apptype" abstract="TRUE">
                <name>Type of Application</name>
                <description>Will the serious game run in a web browser
or/and will it be a standalone application</description>
                <cardinality min="1" max="2"/>
                <feature id="webbased" extends="TRUE">
                  <name>Web-based</name>
                  <description>Playable in a webbrowser. Specify any
limitations or choices</description>
                </feature>
                <feature id="standalone" extends="TRUE">
             <name>Stand-alone application</name>
             <description>Needs to be downloaded and/or
installed</description>
                </feature>
             </feature>
             <feature id="peripherals" abstract="TRUE" optional="TRUE">
                <name>Peripherals</name>
                <description>Use this if you want to use some
peripherals</description>
                <cardinality min="0" max="7"/>
                <feature id="camera" extends="TRUE">
                  <name>Camera</name>
                  <description>Build-in camera of desktop,laptop,
tablet, ...</description>
                  </feature>
                <feature id="microphone" extends="TRUE">
                  <name>Microphone</name>
                </feature>
                <feature id="speakers" extends="TRUE">
                  <name>Speakers</name>
                </feature>
                <feature id="wii" extends="TRUE">
                  <name>Wii</name>
                </feature>
                <feature id="kinect" extends="TRUE">
                  <name>Kinect</name>
                </feature>
                <feature id="hdcamera" extends="TRUE">
                    <name>High Definition Camera</name>
                    <description>External Camera</description>
                </feature>
                 <feature id="sensors" extends="TRUE">
                  <name>Sensors</name>
                  <description>Specify which sensors will be
used/needed</description>
                </feature>
             </feature>
             <feature id="imptechnology" optional="TRUE">
                <name>Implem. Technology</name>
                <description>Use this if you already know which technology
will be used for the implementation</description>
               </feature>
        </feature>
        <feature id="resources">
            <name>Resources</name>
            <description>Describe the resources available for implementing
the game </description>
        <feature id="time">
```

```
          <name>Time</name>
          <description>Indicate the time available to implement the serious
game</description>
         </feature>
        <feature id="budget">
          <name>Budget</name>
          <description>Indicate the budget available to implement the
serious game</description>
        </feature>
        <feature id="people">
          <name>People</name>
          <description>Indicate the fte's available to implement the
serious game</description>
        </feature>
     </feature>
   </feature>
</product>
```

Appendix B – Output of the Pilot Evaluation Session

```
* My Serious Game ( Start here)
The Messenger
 * User Aspects (The game should match the target users/players. Describe them using the given Guideas)
    * Age Range (The age of the player may influence different aspects of the game. Specify the age range here.)
          12-15
    * Girls/Females
    * Player Type  (Use this if you know that the players belong to a particular (or several) player type(s))
      * Type by Personality (This is a classification given by Bartle (1996))
      * Type by Motivation (This is a classification given by Oblinger (2004))
    * Personality Charactersitics (Document any characteristics of the players that could be relevant)
    * Competences (Document any competences of the players that could be relevant)
    * Learning Style (Use this if you know that the users have( a) particular (preferred) learning style(s))
      * VARK Learning Styles (Classification of Fleming)
 * Context of Use (It is important to know in which context(s) the serious game will be used)
    * PC
          Web based, dus via meerdere platforms


    * At School/Institute or In Organization
    * Restricted (The serious game will only beavailable for certain groups)
 * Pedag. Aspects (Describe the pedagogical aspects using the given Guideas)
    * Attitude Change (To change the attitude of the player towards a certain problem/person/issue)
    * Cased-based Learning (Learn by addressing a real world problem/case)
    * Pedagogical Context (Use this to specify if (and how) the serious game will be part of a pedagogical package)
    * Awareness (To raise awareness about a certain problem or issue)
    * Knowledge Acquisition (To increase the knowledge of the player about a certain problem/issue)
    * Practicing Skills (To let the player practice some skills acquired in the game or elsewhere)
    * Social Problem Solving (To learn how to solve a (social) problem(s))
    * Roleplay Simulation (Learners take on the roles of specific characters or organizations in a contrived setting)
 * Game Aspects (Describe the different aspects of the game using the given Guideas)
    * Detective Game (Game in which some mystery must be solved)
    * Single Game (Use this if the game is a single (maybe large) game)
    * Single Player Game (Game played by an individual)
    * Game Concept (Explain the main concept/idea of the game)
          Scenario The Messenger
    * Game Length (Indicate how long it should approximately take to play the game)
          50 min. Per level, meerdere levels mogelijk
    * Game Elements (This part is to indicate what kind of elements you want to have in the game)
      * Dialogs (How will the communication between player(s) and game be supported)
          * Textual (Player is using text)
```

      * Textual (using text (e.g., text ballons))
      * Player to Player (How will the player communicate with another player)
      * Using Speech (Game is using speech)
    * Progress in the Game (Rewards is by allowing the player to progress towards the goal)
    * Progress Indicators
    * Setback (Player is set back to a previous point in the game)
    * Self Reflection (Allow the player to do some kind of self-reflection)
    * Buddy (Use this if you want to give the player a buddy (helper))
      * Give Advice (Buddy will give advice)
      * Pedagogical Agent (The buddy is more like a teacher/coach)
    * Avatar (Use this if you want to represent your player(s) by avatar(s))
      * Iconic Character (Will usually be related to the story/background of the game)
      * Similar Age (Age should be similar to age of the player)
      * Similar Gender (Gender should be similar to gender of the player)
    * Extras (Rewards are in the form of extra elements, new abilities)
    * Points (Player earns points)
    * Enabling/Disabling Actions or Interaction
    * Advices or Guidance
  * Simulation Game (Game that simulate aspects of a real or fictional reality (such as management, life, flying, etc))
  * Role Playing Game (Players assume the roles of characters in a fictional setting)
* Implem. Aspects (Describe the implementation aspects using the given Guideas)
  * Web-based (Playable in a webbrowser. Specify any limitations or choices)
  * Speakers
  * Implem. Technology (Use this if you already know which technology will be used for the implementation)
      Game engine howest
Attac- I
* Resources (Describe the resources available for implementing the game)
  * Time (Indicate the time available to implement the serious game)
      2012-2016
Demo 1: feb 2015
Eerste labtests
Demo 2: juni 2015

  * Budget (Indicate the budget available to implement the serious game)
      Context van iwt sbo project.
Specifieke delen van budget voor scenario, grafische vormgeving...
  * People (Indicate the fte's available to implement the serious game)
      Ua: 2 fte, 1 20%
Vub: 2 fte
Ugent: 1fte
Howest: 1 fte