TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

MASTERARBEIT

A Generic Software Template to support Learning on Mobile Devices

Ausgeführt am Institut für
Softwaretechnik und interaktive Systeme
der Technischen Universität Wien
unter der Anleitung von
Ao.Univ.Prof. Mag. Dr. Silvia Miksch
durch

Bakk. techn. Andreas Böhme

Sonnleithnergasse 15/3/4/50
A-1100 Wien

Wien, am 18. Dezember 2007
Datum

Unterschrift (Student)

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Wien, 18. Dezember 2007

Unterschrift

Kurzfassung

Nahezu jeder Mensch besitzt heutzutage ein Mobiltelefon, und hat es wohl den ganzen Tag bei sich. Was wäre naheliegender, es nicht nur als Telefon zu verwenden, sondern auch um unterwegs z.B. Nachrichten zu lesen oder Filme zu sehen. Hersteller von Handys kamen zur gleichen Schlussfolgerung, und begannen sogenannte *Smart Phones* herzustellen. Diese werden standardmäßig mit E-Mail Clients, Web-Browsern, Multimedia-Playern, sowie einigen anderen nützlichen Anwendungen ausgestattet.

Auf dieser Idee aufbauend, stellt diese Arbeit ein Konzept zum Lernen unter Verwendung von Mobiltelefonen dar. Basierend auf aktuellen Forschungsergebnissen im Bereich des mobilen, sowie des Spiel-basierten Lernens, möchte ich hier einen Ansatz vorstellen, der die beiden Forschungsbereiche kombiniert. Es werden dabei Lernstrategien mit Spaßelementen verbunden, um vor allem die Zielgruppe von 18-24 jährigen StudentInnen anzusprechen. Die zugrunde liegenden Forschungsergebnisse, sowie die gesamte System-Architektur werden ausreichend dargestellt, das Hauptaugenmerk liegt jedoch auf der technischen Spezifizierung und Implementierung der mobilen Anwendung.

Abstract

Nowadays almost everybody owns a mobile phone and carries it around the whole day. It is obvious, though, to use it not only as a phone, but also as a tool to do other things, like reading news or watching videos while travelling from or to e.g. work or school. Cell phone manufacturers acted on that idea, by producing so called *Smart Phones*, that are equipped with e-mail clients, web-browsers and multimedia-players, beside other useful applications.

Going a step further, this work presents the concept of learning with someone's mobile- or smart-phone. Based on state of the art research in the fields of mobile learning and game-based learning, I am presenting the implementation of a mobile game-based learning approach. It incorporates learning strategies with fun elements to make it more attractive to the target group of students at the age of 18-24 years. While describing research basics, as well as the whole system-architecture, the work focuses on the technical specification and implementation of the mobile application.

Acknowledgements

My greatest debts are owed to both my family and my girlfriend, who have always and tirelessly supported me for my whole study, and beyond.

Furthermore, I want to thank all my colleagues of the Studio Smart Agent Technologies, SAT (which is part of the Research Studios Austria), as well as all other participants of the EU project *Mobile Game-Based Learning* (*mGBL*). Special thanks, however, go to Ulf Harr (SAT) who designed the authoring interface (appendix C) as well as the controls graphic (figure 5.5), Erik Fürst (SAT) for creating the content images (figure B.2) and Brigitte Krenn (SAT) for fruitful discussions. Moreover, I want to thank Florian Koch (Sensomatic) who created the design elements used in the client-application (e.g. the boss images), Marko Justinek (University of Maribor) for providing the basic version of figure 5.2, as well as Alice Mitchel (Inspire) who created the majority of the presented content of the learning games. The images of the mobile phones (figures D.1 to D.4) are taken from the manufacturers' websites. Parts of this master thesis were incorporated in *mGBL* deliverables and publications.

Special thank also goes to my carer Prof. Mag. Dr. Silvia Miksch, who supported me continuously with precise and helpful advice.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem

Nowadays information is very important for nearly everybody. Take *Wikipedia*[1] as an example. It covers 1.47 million articles, and grows per approximately 1820 every day[2], while the library is consulted more than 6 Million times a day[3].

On the one hand, we simply don't know how to keep up to date with masses of information, that may come from newspapers, books, scientific papers or *just* the Internet. And we all know the problem of not using already gathered wisdom regularly. That is, what makes us lose it piece by piece.

On the other hand, knowledge builds the basis for decision making. But knowledge is not the only factor – the other is practice. Especially in life-threatening areas (e.g. hospitals' emergency rooms), it is essential to make a decision in a short amount of time.

Therefore, it is necessary to create a space, where people can practice, and get feedback about the quality of their choices, but without the fatal consequences of the daily life.

1.2 Goals

The overall goal is to improve the effectiveness and efficiency of learning, as well as decision making, in a cognitive and emotional way. To accomplish this overall goal, it can further be divided into the following sub-goals:

- Develop (innovative) learning models that can be applied to different learning domains.
- Specify and implement, at least prototypical, games, based on the above mentioned learning models.
- Examine those learning models in user tests to derive their effectiveness and efficiency.

The EU project *Mobile Game-Based Learning* (*mGBL*)[4] has been originated to address exactly the above defined goals by developing educational game templates for mobile phones. A server side tool is used by teachers/game authors to create several game instances by combining a game template with different contents.

The following list sketches the basic steps defined by *mGBL*:

[1]`http://en.wikipedia.org`, accessed at 10.10.2007
[2]New English articles in October 2006 [Zacb].
[3]Accessed English articles in October 2004, which is the last recorded date in [Zaca].
[4]For more details concerning the project see chapter 4

1. Plan and carry out a user-requirements analysis for revealing the needs of all stakeholders (teachers, students, IT-staff, etc.).
2. Consider and choose, or create pedagogical frameworks and learning models for supporting students in developing learning and decision making skills.
3. Design and implement a prototype server platform and client software.
4. Plan and carry out user-trials of the implemented software, in a way that allows to verify the effectiveness of the developed learning models.

The above listed goals apply to this work, too, although differently weighted. For example, *mGBL* demands for implementing a server platform and at least three game templates, while this work focuses on the one game template, it is based on. For more details about the difference of the *mGBL* project and this work, please see section 4.6. However, the next chapter explains the organisation of this work, and makes clear how the chapters are weighted.

1.3 Structure of the Work

Starting with state of the art research in the field of learning, the first chapter (chapter 2) explains the benefits, and therefore reasons for choosing mobile learning as well as game-based learning. The following chapter (chapter 3) lists the most recent development technologies, their benefits and short comes, and goes into more detail with the technology, chosen for this project (*JavaME*). The chapters 5 and 6 build the main part of this work, because they describe the specification and implementation details of *mGBL*'s game template 1 *Ahead of the Game*. You get insights about the two components of the template, quiz and simulation, how they could be combined, and their interaction with the outside world. A detailed insight about the implementation, gives a better understanding of how the application can be used, or extended with new ideas. At the end of this work I present findings of the conducted user trials, as well as ideas for further improvements (chapter 7).

Chapter 2

Learning Theories

We have moved from "Learning for life", which is limited to the period when we were pupils or students, to "Lifelong Learning", a process accompanying us for the rest of our lives.

We also have moved from teaching to learning and from authoritarian to authority in education. Formal education plays a limited part in our life, and we all know that people learn a lot from TV, newspapers, the Internet and other sources [PKD04]. We are now at a point where the society accepts (even slowly) that game-based approaches or additions in education can lead to good results — or even better ones than traditional approaches. We also know, that we are living in a fast moving world where time is limited, and we have to make the best of our opportunities and use the time while travelling to work or school for useful tasks like learning. That's why *game-based learning* and *mobile learning* as well as the combination of both areas (*mobile game-based learning*) have a very high potential to expand, and thus improve, our well known educational systems.

2.1 Formal, Non-Formal and Informal Learning

The government needs formal assessment of individuals, where companies and organisations are interested in both formal and non-formal education for their employees to denounce their new skills and knowledge. But, what happens behind the scenes, by cooperating with other people, is informal learning (please find exact definitions below).

Sweden, for example, realized that non-formal and informal learning is very important for the development of individuals, and is therefore accredited by the government [PKD04].

Formal Learning takes place in an education or training institution and is well structured which means that the objectives and the time frame for learning are clearly defined and there is some kind of learning support – a teacher or guide – available. It is intentional from the learner's perspective. Formal learning requires measurable knowledge and skills and is therefore demanded by state, because otherwise assessment seems not possible.

Non-formal Learning takes place aside the mainstream system of education and training and does typically not lead to formal certificates. It can be provided in the workplace, through activities or civil organisations or political parties. Like formal learning, it is also intentional from the learner's perspective. Non-formal learning does not verify skills and knowledge, but rather makes it visible to the learner selves and others. It is characterized by how, not which, gathered skills and knowledge are used.

Informal Learning is a natural aspect of everyday's life. Unlike others, informal learning is not necessarily intentional from the learner's perspective. It, therefore, may not be recognized by individuals themselves. Because it is most often unintentional and not recognized, informal learning is neither structured, nor does it lead to certification.

2.2 Game-based Learning

"Education is what remains, when what was learned has been forgotten." [PKD04]

Therefore, education should be the active engagement of learners with the subject matter, instead of its passive reception.

Learning through games is a concept that has always existed in the human history. Children playing with each other and young adults playing computer games have many in common — motivation is high, and they can play for hours even not realizing that time passes by. That is exactly what teachers wish in their own education.

Pivec et al. [PKD04] describe an example of a medical student, who is "playing being a doctor", where in the early stages of education – maybe even before the education starts – this play has indeed a large element of fantasy in it. But this fantasy should not be viewed negatively, e.g. as self-delusion, but rather positively, as high aspiration. During his education the medical student is increasingly asked to play the role of a medical practitioner, where in this role his skills and knowledge become more and more cultivated. The role-play is thus used not only as a teaching method in medical education, but also as a method of assessment. The process by which the behaviour of the student ends to be a role-play and starts being reality is a gradual and often invisible one. Safety is (beside trust) an essential element of the social contract of a game. It is a prerequisite for a player to be comfortable enough to dunk into the game and it's social context [SZ03]. Obviously, there is nothing trivial or frivolous about these activities that are orchestrated by medical educators, nor should it be in other areas of education.

2.2.1 Game Definition

A game is...

> "A contest with rules to determine a winner." [Wor]
>
> "A pursuit or activity with rules performed either alone or with others, for the purpose of entertainment." [Wikk]
>
> "Reduced to its formal essence, a game is an activity among two or more independent decision-makers seeking to achieve their objectives in some limiting context. A more conventional definition would say that a game is a context with rules among adversaries trying to win objectives." [Abt87], cited in [SZ03]
>
> "Any overt instructional or learning format that involves competition and is rule guided." [DRL96], cited in [PKD04]

Summing up these definitions, a game consists of rules, activities, players or decision-makers, a winner or winners, and is set up in a context where a competition takes place. There may be more elements to be found in other definitions, which may raise the complexity of a game or the understanding of games. However, those elements must be considered when defining a new game (please find further information on defining a game in [SZ03]).

2.2.2 Games and Computer Games

Research on the potential of computer games in educational contexts dates from the earliest days of the microcomputer ([Mal81a, Mal81b], cited in [PKD04]).

The key advantage of computer games over conventional games is their knowledge about the status of the player and the game itself. Motivation is generated by solving challenging but achievable tasks — beside active participation, specific and prompt feedback and some kind of uncertainty. Based on the current game status, the program is able to continuously motivate players by analysing their competence-level and adopt the difficulty accordingly. This way, the software can start with a difficulty level the player is comfortable with, and increase it progressively to set it always a bit beyond the player's comfort level. Thus, by providing maximum excitement, the player is always moving forward at an optimal learning rate. [PKD04]

2.2.3 Benefits of Game-based Learning

The most important benefits of game-based learning are the learner's motivation and engagement, here motivation is considered as one of the key factors contributing to effective learning. Having fun is an additional factor that is central for effective learning. [PKD04]

To achieve the learner's motivation, games do not have to be novel, or challenge them (apparently) with the usage of complicated or sophisticated equipment. Such games rather provide players with continuous emotional conditions or psychological stimuli. These conditions ideally vary during the game play and might include satisfaction, desire, anger, interest, excitement, enjoyment, pride of achievement, peer recognition, and many more.

Compared to traditional learning formats, video and computer games create a unique form of "play environments" where motivation and fun provide a powerful learning experience. The player, irrespective of age, is acting in an active and performance-based play environment, including all related activities such as active discovery, analysis, interpretation and problem-solving, sometimes supplemented by physical activities.

Game-based learning approaches, help people to develop skills, that are necessary to transfer information and skills from one situational context to another. To achieve this, coherence between the game and real life contexts is extremely important.

Security is another great advantage of games, and especially digital games, which enable learners to act in a risk free virtual environment without the sometimes fatal consequences of real life. Players have the opportunity to make (wrong) decisions, evaluate the consequences, and refine their judgement and decision making process.

Finally it should be said that it may be more effective to use selected elements of a game that seem appropriate, rather than the game as a whole, in a curriculum. [PKD04]

2.2.4 Selecting or Finding a Game

The first task in finding an appropriate game to support learning is to identify the educational objectives that should be achieved.

Pivec et al. define three basic approaches for selecting a game. The *mGBL* project adds another one:

- Use an existing – commercial or non-commercial – game and embed it in the education process. Games like *SimCity*[1] may be used for example in the context of ecology or resource management. In contrast, simulation games like *The Sims*[2] or *Second Life*[3] may rather be used for social and behavioural topics. Several other commercial games may be used for learning areas like history, maths, logistics, and others.

- Use an existing game that has been specifically developed for an educational purpose. The most educational games can be found in the areas of business studies and political science. [ASS71, Pre01] cited in [PKD04]

- Build a new game for the desired purpose. As you might know there exist some freely available, as well as commercial, tools that help developing a computer game. As an example think of the *Unreal Engine* that originally has been created for the Ego-Shooter *Unreal* and was used in many other games so far [Wikj]. While commercial games cost about 1 million USD to 50 million USD to bring to the market place where the costs of 350.000 USD for the Unreal Engine are legitimate, there is a much lesser budget for educational contexts, what makes such engines less attractive [Wikj].

- Adapt a game template, or some other kind of skeleton, that can easily be modified to the author's needs. This seems to be the most attractive one of the creational approaches, because it combines a manageable implementation effort at limited costs with the potential of individual and creative fabrication of professional games.

2.2.5 Characteristics in the Learning Setting

Some key characteristics of the course should be taken into account for the integration of a game-based approach. Some of the most important characteristics are [PKD04]:

- the size of the students' group — while some activities are only possible with a certain number of students, a large group size may, in certain situations, create logistical problems.

- the information and communication technology skills of the students group

- the technical resources which are available for the student group to use — hard and software must be accessible to everyone who wants, or needs, to use it.

- the technical and educational support, the teacher can count on

[1]`http://simcity.ea.com/index.php?`, accessed at 10.10.2007
[2]`http://thesims.ea.com/`, accessed at 10.10.2007
[3]`http://secondlife.com/`, accessed at 10.10.2007

2.2.6 Introducing a digital game into a course

Game-based approaches will usually be included in a course as one of several elements. Its aim will differ from course to course. It may be one of the following [PKD04]:

- Usage as introduction to a certain topic or activity to trigger further and deeper engagement with the topic. Students often want to know facts behind their favourite computer games, what makes it ideal to introduce, for example, a History lesson with the game *Age of Empires*[4].
- Increase – at least the likelihood of – student participation in formative assessment. The motivation of awarding a grade is often not high in classes, but can be increased by challenges through competition or time pressure. The competition might be between individuals, teams, or by competing with the student's own "best performance".
- A game-based approach may create a simulation environment where students have a safe place to practice in a harmless way. The benefits of such simulations can often not be provided otherwise, what makes it ideal for professional areas like military, medicine, nursing, surgery, teaching and engineering.

2.2.7 Debriefing

To provide the maximal educational benefit, an explicit debriefing may be necessary for game-based learning approaches. [PKD04]

2.2.8 Beyond Education

Like computer technology holds a fascination for some people, encouraging them to engage with the technology behind the scene, starting programming, we can hope that a similar fascination happens to students who are not that technological affine, digging into a topic presented in a game-based approach.

2.3 Mobile Learning

> "Mobile learning is unique in that it allows truly anywhere, anytime, personalised learning." [Att04]

Mobile-Learning (M-Learning) is a new approach to combine the mostly static *Electronic-Learning* (E-Learning) with the power of mobile information and communication technologies. Associated with this topic are challenges in the areas of didactics, technology and organization of the learning process, business models are, in particular, very important, to show non-enthusiasts the power of new learning approaches. Additionally we must keep in minds, that M-Learning is very different to E-Learning, and direct transportation of content from the latter to the first area won't work for the most cases [SHB04]. Leon Rodin therefore started a research with the aim of converting E-Learning to M-Learning content, which concluded that a complete re-work of the content is necessary, otherwise it won't work [Rod04].

[4]`http://www.microsoft.com/games/pc/age3.aspx`, accessed at 10.10.2007

2.3.1 Benefits

Jill Attewell [Att04] identifies some key benefits that come from mobile learning, which have been observed during the *M-Learning*[5] project.

Mobile learning helped learners to improve their literacy and numeracy skills. That mainly comes from the fact that teachers recognized already existing abilities for the first time. One mentor even reported that a pupil improved reading skills just because of switching the media. Because of negative educational experiences he refused reading books, but he indeed read texts and information regarding the device very well.

Mobile learning can be used to encourage both independent and collaborative learning. Some people are glad about the opportunity to work at home at night, when they don't feel under pressure or under control. Others like working together in teams, because it incites to achieve a common goal, or something useful for the group. M-Learning can encourage both, depending on how it is used.

Mobile learning helps learners to remain more focused for longer periods. A mentor reported that a group of pupils was focused up to two hours, where, normally, it was difficult to keep them focused for 15 minutes. Attewell admits that this could be a result of the novelty of using mobile phones for educational purposes. Whether this is true or not will become clearer over time.

Mobile learning helps to raise self-esteem and self-confidence. Both are not related to the learning material, but to the learners' general attitude. While self-esteem has been reported to come from the teachers' trust giving expensive and sophisticated equipment to pupils, the raised self-confidence seems to be an effect of the trial-and-error principle of most electronic devices.

2.3.2 Challenges

Wikipedia identifies challenges in the field of mobile learning that are divided into two categories [Wikg]. Based on these, I enhanced the two lists with experiences due to my work with mobile phones, and findings of research papers according to mobile learning.

Technical challenges include:

- Connectivity — Wap, GPRS, Bluetooth, WLAN, etc. can be used for interacting with other devices. But not all phones are equipped with all these technologies, what makes it essential to support a few of them and making it possible to select one's preferred technology.
- Battery life — stand-by times of modern devices are up to one or two weeks, though this shortens to a few hours when the device is in use. Especially when the mobile phone is used for purposes it was intended for, this could let battery power of a few minutes for other activities like learning. Therefore it is essential that mobile learning

[5]`http://www.m-learning.org`, last accesses on 10.10.2007

applications support small lecture chunks, that can easily and automatically be saved in short intervals.

- Interacting with small devices — it's a challenging task to design a user interface for mobile phones (which usually lack of the input devices available for computers) that suits to many different devices allowing to control them in an easy and intuitive way.
- Displaying useful content on small-screen devices — content has to be adopted to the small screens with low resolutions that are typical for mobile devices (in average such screens are about 5cm large and have a resolution of 176x208 to 240x320 pixels).
- Presenting different kinds of media — due to the vast variety of used technology, and the (often) low processing power of current mobile phones, it is often hard to find a common ground for multi-media content (video- and audio-snippets, office files like Word, Excel, PDF).

Social and educational challenges include:

- Assessment of learning outside the classroom — we know from [Att04] that mobile learning helps to recognise existing abilities, and assessment should accommodate that, even though as just one facet in the assessment process.
- Tracking of results and proper use of this information — the results can be used in different ways, like for assessment, or for constructive work that banks on these outcomes.
- Support of learning across many contexts — especially with reduced budget that doesn't make it possible to develop discrete mobile applications for each scope.
- Developing an appropriate theory of learning for the mobile age — this learning model ideally makes use of the potential of different available technologies (digital cameras, GPS-Navigation, etc.), by combining them into one learning scenario. This makes the education approach more attractive for younger learners, and introduces variety for people of every age to keep them motivated with the topic.
- Develop and preserve ethical principles like confidentiality of data, anonymity of participants or a *Netiquette*, similar to those we are already familiar with, from e-mail or chat. [TB04]

2.4 Mobile Game-based Learning

Mobile game-based learning is the effective combination of M-Learning with elements of game-based learning to produce appealing and motivating learning software.

Graham Brown-Martin describes an opportunity we now have at schools:

> "Actually, we don't have a choice. Young learners already have this technology [mobile phones], and for the very first time schools don't have to provide that. They just have to embrace it in a meaningful and useful way." [Mat]

This opportunity can easily be generalized to many other formal, non-formal and informal learning environments, and offers great potential in these educational areas.

Chapter 3

Mobile Applications

Some of the most important technologies for developing applications or games for mobile devices are: *J2ME*[1], *Flash Lite*[2], *.Net Compact*[3] and *C++* (for Symbian OS)[4]. The former two are independent of the operating system, but still require some kind of interpreter on the host system, the latter two are built upon the APIs of specific operating systems, and can therefore only be executed in these environments. Java is the most common and most supported programming language for mobile devices, where C++ became more popular in the last few years, because Symbian OS gained a larger market segment as operating system for cell- and smart-phones. Dot-Net Compact, which can only be used on Windows CE powered devices, and Flash Lite are the least popular of the here presented technologies.

All technologies have some common ground, that is, they have to cope with reduced processing power, memory, and small, differently sized displays. But each has its advantages, like better performance (C++), easy graphics support (Flash Lite) or a wide distribution (*JavaME*). However, the used technology depends, like it does for developing desktop applications, on the specific use cases.

This chapter presents some characteristics that should be kept in mind when developing software for mobile devices (especially mobile phones), and concludes with an overview of the above listed technologies. It focuses on *JavaME*, because it is the technology used to implement *mGBL*'s game template 1 *Ahead of the Game*.

3.1 Specialities of SW-Development

When you start working with *JavaME*, you might think that there is no difference to developing desktop or server applications using Java. This could be a great mistake.

A blue-eyed *JavaME* development project may proceed like this:

1. You design the software in the object-oriented manner, you are used to.
2. Implementation starts, and you find out that you have to be very creative in replacing expected API functionality, that is not available for *JavaME*.
3. Bugging around with creating the desired GUI.

[1]`http://java.sun.com/javame/index.jsp`, accessed at 10.10.2007

[2]`http://www.adobe.com/products/flashlite/`, accessed at 10.10.2007

[3]`http://www.microsoft.com/germany/msdn/library/mobility/windowsmobile/pocketpc/DasNETCompactFramework.mspx`, accessed at 10.10.2007

[4]`http://www.symbian.com/`, accessed at 10.10.2007

4. It is necessary to shrink used resources (e.g. images) to undergo the file size limits of the mobile device that's used for early tests.

5. Early tests show that the application reacts very slowly on user input, what leads you to get familiar with emulator-based debugging and on-device-debugging.

6. Mastered points 1 to 5, you want to show the application to a friend and transfer it on his/her device, where:
   - it does not work at all
   - it looks completely different
   - the application crashes arbitrarily

7. At last you end up in adopting the application to various mobile phones which doubles your previous efforts.

Though, this scenario describes the worst case, it shows that it is very essential not to underestimate a "small Java application". Mostly you won't be able to finish the project; it would take you nearly the whole planned project time to implement only half of the above sketched tasks, which do not even include changes in the software itself, e.g. change requests, or changes in the software architecture.

To circumvent such disastrous projects, you may want to get familiar with some basics, that are special to mobile devices and their software development. Based on this knowledge, it is easier to estimate a software development project plan.

3.1.1 Limitations

Often people are not aware, that mobile phones are not as powerful as PCs, and expect that the same "software magic" is possible. Here are some important limitations, listing the more obvious ones first:

- The *screen size* is limited and different for many devices. MIDP specifies a minimum of 96x54(!) pixels [Sch07]. Displays of modern mobile phones have a resolution that lies between 128x128 and 240x320 pixels. That kind of variations is typical for mobile devices, and very unique compared to PC monitors and their resolutions, for example.

- Usually a cell phone has *a joystick* and *12 keys* (10 for the numbers 0 to 9 and 2 soft keys) to navigate through menus and dial numbers[5]. A computer keyboard has 104 or even more keys and is supported by a pointing device (usually a 3-button mouse).

- The *processor* is more powerful than the one used for travelling to the moon was, but that is still only about 1-15 percent of modern desktop CPUs.

- An obvious size is the *storage capacity* of a mobile phone, which often is expandable. But even more important is the available *RAM size*, which specifies the maximum transient memory capacity, and the heap-size, that further reduces the available RAM size for one instance of a (Java) Virtual Machine.

[5] Smart Phones are equipped with a full keyboard and a stylus for pointing tasks, but the interaction is still different combined to PCs, and you can't expect whether the user prefers the one input method (keys) or the other (touch-screen).

- Although, the *permanent memory* of modern mobile phones is large, some devices limit the size of .jar files (e.g. Nokia's 6230i accept only .jar files that are not larger than 512KB).
- *Access to the Internet* is often slow and expensive and the user needs to allow it (for exactly those reasons).
- Playback of *multimedia files*, like MP3 or video snippets, works quite well now, but is often limited to a few selected encoding-formats that differ from one to another device. Streaming is often not an option, because there is no standardized way to do that on mobile phones, why own algorithms have to be implemented, which leads often to bad results because of the devices' reduced processing power.
- Access to installed applications is often prohibited. It is not possible to (PC-like) drag&drop a number from the phone book to another application.
- The *API* of programming languages is limited due to the reduced RAM and CPU resources.
- Last but not least: *there are huge differences from one device to another.*

3.1.2 The Software Development Process

When you are developing software for mobile devices, you will experience that the software development process is quite different to desktop or server applications.

It starts with the *software design*, that needs to be less object-oriented to improve the performance [ZHKH07]. That means you have to find the correct balance between object-orientation (to benefit from readability, re-usability, etc.) and performance, on the target device. The more classes you have in your application, or the more instances you create, the slower it will get.

The *implementation* suffers from missing APIs, developers are used to from their work with desktop applications. This concerns not only high level GUI components, but also XML-Parser, or even logging APIs.

Source code compilation is not one task but consists of pre-processing (optional), compilation, obfuscation (optional), pre-verification, packaging, signing (optional) and jad-creation[6].

For *debugging* purposes emulators might help in the first instance, but can't replace a real device. And: because you don't have access to the underlying operating system, on-device-debugging is not an easy task. Some cell phone vendors and associated companies provide special development kits for on-device-debugging, unfortunately they don't work for all devices.

The *adaptation to different devices* may take up to 90% of the overall project time and costs, depending on the quantity of supported devices, the complexity of the GUI design, the used APIs and maybe some other factors. To reduce the efforts, or make them just less surprisingly, it may help to organize the adaption to different devices as a parallel task to the whole development process, like it is often done for quality assurance.

[6]These steps describe the *JavaME* build-process. The compilation process of other programming languages for mobile devices may be less complicated. However, knowing the process might help preventing troubles.

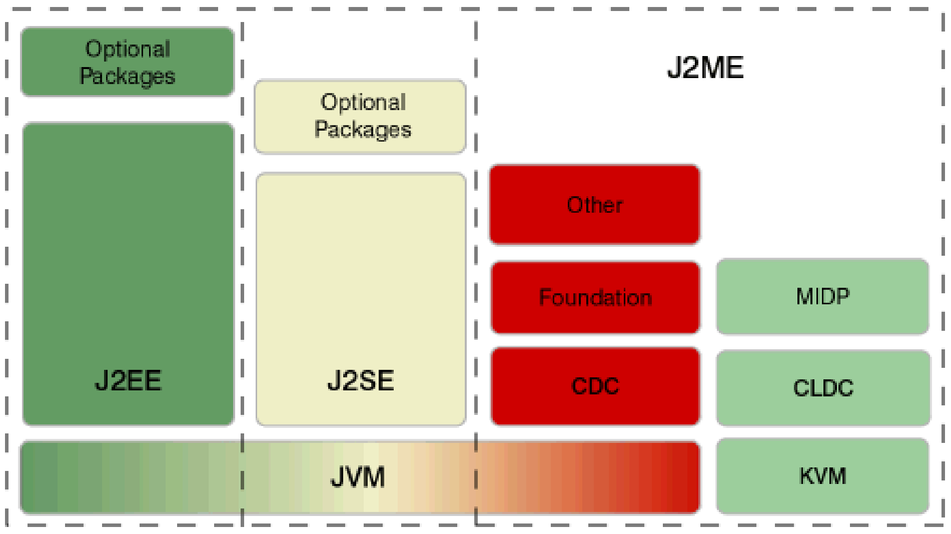


Figure 3.1: The Java family at a glance [Sun01]

3.2 Technology

3.2.1 J2ME

Java was invented in the year 1996 by Sun Microsystems and continually gained popularity, especially for desktop applications[7]. That's the reason, why Sun made no efforts to keep the resource demands low. When mobile devices become more popular, a subset of J2SE version 1.1 was used to build the basic J2ME API upon. Some classes where changed, some added[8], but most of them have been dropped. The result was a *Java Virtual Machine* (JVM) that uses only some kilobyte of memory, and therefore was called the *Kilobyte Virtual Machine* (KVM). Figure 3.1 shows an overview of the Java family including J2ME, J2SE, and J2EE. Additionally, figure 3.2 shows how different the mobile specifications (CLDC, CDC)[9] are from their desktop pendant.

Architecture

As for every device, the operating system is the interface to the device's hardware it is built upon. Based on the operating system, *JavaME* defines several layers that are arranged one above another (figure 3.3, [Sch07]):

- a configuration
- profiles

[7]The current version for desktop applications is 1.7 (specification started in August 2006, planned release is 2008) [Wikd].

[8]For additional classes the standard package `java.*` can't be used, because it contradicts the basic Java specification. For those classes the root package `javax.microedition.*` is used.

[9]The *Connected, Limited Device Configuration* (CLDC) and *Connected Device Configuration* (CDC) are described in more detail on the following page

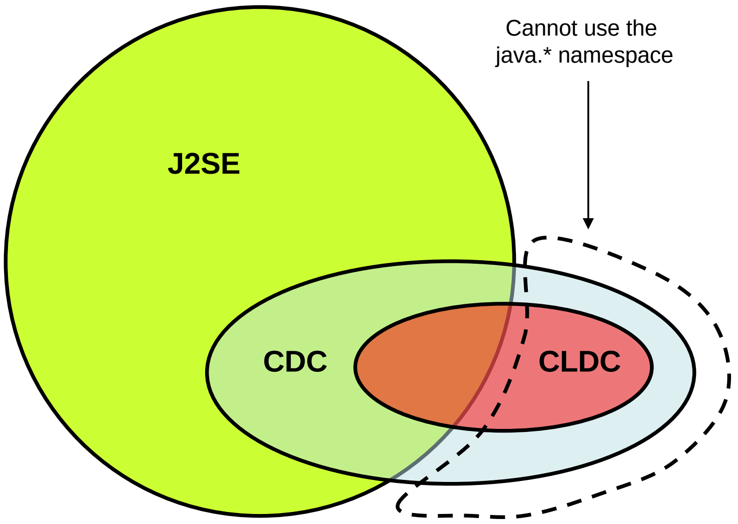


Figure 3.2: A schematic presentation of the relation between J2SE, CDC and CLDC

- optional packages

Configuration

The configuration defines minimal requirements to the host platform for a set of similar devices with identical or similar resources like screen size, CPU power, or the ability for floating point calculations – i.e. a horizontal class of devices. It further specifies the supported language scope (e.g. Generics[10] are not supported), the JVM's supported functional scope (e.g. the on-the-fly verification[11] is not supported; find further information on page 17), and the minimal set of library classes. Although the term *Java* in *Java Micro Edition* suggests that someone can use the full power of Java on mobile devices, only a subset of the language is supported.

Currently two configurations are in use[12]:

- The *Connected, Limited Device Configuration* (CLDC) is used for small mobile devices like cell phones.
- The *Connected Device Configuration* (CDC) is best known to be used for more powerful mobile devices like PDA's, but is intended to be used also for e.g. Set-Top-Boxes.

Version 1.1 of the CLDC specification, which was finally released in the year 2003, brought some benefits in contrast to its predecessor CLDC v1.0, for example:

- Floating point support; previously only integer values were supported

[10]Find a definition here: `http://en.wikipedia.org/wiki/Generics_in_Java`, accessed at 10.10.2007
[11]The Java verification process is described here: `http://en.wikipedia.org/wiki/Java_Virtual_Machine#Bytecode_verifier`, accessed at 10.10.2007
[12]Nowadays so called Smart-Phones are very popular, which combine the functionalities of a cellular phone and a PDA and therefore support both configurations.

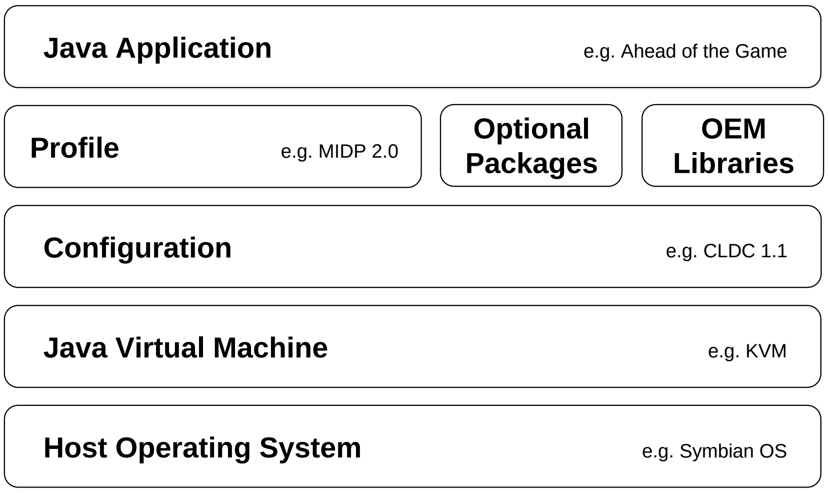


Figure 3.3: *JavaME* software layer stack.

- Weak reference support; a weak reference allows the garbage collector to remove a referenced object to save memory, where it can easily be restored, when needed.
- Time relevant classes (namely: `Calendar`, `Date`, `TimeZone`) have been redesigned to be more compliant to the J2SE standard.
- Threads became more compliant to the J2SE specification (the name of a thread has been introduced, as well as some new constructors).
- The minimum memory budget has been raised from 160 to 192 Kilobytes, mainly because of the introduction of floating point arithmetic.

Profile

A profile extends the configuration with more specific functionality like file-storage or visual components — i.e. a range of functions of a vertical market segment. In contrast to a configuration, a profile may specify optional units (that must not be implemented by vendors). The most important profile is the *Mobile Information Device Profile* (MIDP), which was initially released in September 2000, and has been enhanced to version 2.0 in November 2002.

Optional Packages

Optional packages are defined in the *Java Community Process* (JCP), together with all relevant vendors, although they are not liable for implementing them. Using optional packages enables software engineers to write code against defined interfaces of a horizontal class of devices, reducing configuration and/or implementation effort. Some popular, optional packages are shown in table 3.1, together with the above described configurations and profiles, to draw a complete picture.

Table 3.1: Overview of the most important configurations, profiles, and some selected optional packages for *JavaME* [Com].

JSR[13]	Package	Description
30, 139	CLDC, versions 1.0 and 1.1	Defines a standard platform configuration of the *Java^{TM}2 platform, Micro Edition* (J2ME$^{\text{TM}}$) for small, resource-limited, connected devices.
37, 118	MIDP, versions 1.0 and 2.0	Defines a profile that extends and enhances the *J2ME^{TM}Connected, Limited Device Configuration*, enabling application development for mobile information appliances and voice communication devices.
82	Java APIs for Bluetooth	Bluetooth is an important emerging standard for wireless integration of small devices. The specification standardizes a set of Java APIs to allow Java-enabled devices to integrate into a Bluetooth environment.
120, 205	Wireless Messaging API (WMA), versions 1.1 and 2.0	Defines a set of optional APIs which provides standard access to wireless communication resources (like SMS or HTTP), designed to run on J2ME configurations and to enhance J2ME profiles with unique functionality.
135	Mobile Media API (MMAPI)	Specifies a small-footprint multimedia API for J2ME$^{\text{TM}}$, allowing simple, easy access and control of basic audio and multimedia resources while also addressing scalability and support of more sophisticated features.
179	Location API	Enables developers to write mobile location-based applications for resource-limited devices. (The API works on the J2ME CLDC v1.1 and CDC configurations.)
226	Scalable 2D Vector Graphics API	Defines an API for rendering scalable 2D vector graphics, including image files in W3C Scalable Vector Graphics (SVG) format.

OEM-Specific Libraries

Often a vendor wants to lay more power in the hands of developers, by letting them access functionality via the programming language, that is already implemented with his/her devices. Otherwise, it may be that there is no JSR specified or completed for a particular functionality, and therefore no standardized programmable interface is publicly available, yet. In these cases

[13]The *Java Specification Request* (JSR) is an integral element of the *Java Community Process* (JCP), which was founded to build a platform for extensions of the Java technology. A JSR is a technical proposal for extensions, where the final version of a JSR requires a reference implementation. [Wikc]

a vendor may define his/her own libraries for a hand of specific devices, or – more often – a certain product line (e.g. Nokia's Series 60). It is obvious that those libraries only work for the specified target devices. A good example is *Nokia's UI API*[14], which extends the MIDP 1.0 profile with transparency effects, polygons (e.g. triangles), image processing capabilities (e.g. rotating), as well as support for vibration and backlight controls.

Preverification

For security reasons, the Java Standard Edition verifies the bytecode of each class that is to be executed, when the application is about to be started [Wike]. This ensures, for example, memory protection and avoids the execution of invalid instructions. Usually this process requires a fair amount of heap-size and CPU power (and therefore time).

In the past this was no option for small devices like mobile phones, and is still deprecated. Therefore a pre-verification procedure has been developed that transfers most of the time and memory consuming procedures to the machine where the source-code compilation is done, thus a PC.

Pre-Verification is a two-fold procedure:

1. The pre-verifier processes a given *.class* file in the way, so that it remains semantically the same, but can be verified easily on the target machine.

2. The runtime-verifier does the remaining checks, before the *.class* file is interpreted by the virtual machine on the target device.

MIDlet and MIDlet-Suite

> "A MIDlet requires a device that implements Java ME, MIDP to run. Like other Java programs, MIDlets have a "compile once, run anywhere" potential. [...] MIDlet distributions also consist of a .jad file describing the contents of the .jar file." [Wikf]

A MIDlet has to fulfil the following requirements in order to run on a mobile phone:

- The main class needs to be a subclass of javax.microedition.midlet.MIDlet
- The MIDlet needs to be packed inside a .jar file (e.g. by using Sun's jar-tool)
- The .jar file needs to be pre-verified.
- In some cases, the .jar file needs to be signed by the mobile phone's carrier.

A MIDlet, therefore, can be considered as an application.

A MIDlet-Suite bundles one or more MIDlets into one .jar file, which is then distributed.

A .jar file is basically a .zip file with another file-extension, that incorporates the application's classes, a descriptive Manifest file, resources (e.g. graphics) and the used libraries (e.g. a XML-Parser library) [Wikb].

[14]`http://www.forum.nokia.com/info/sw.nokia.com/id/1f75c9a5-3aeb-42e9-890d-f06e3b2e94d9/Nokia_UI_API_Programmers_Guide_v1_1_en.pdf.html`, accessed at 10.10.2007

The .jad file, that always comes along with a .jar, is basically a copy of the Manifest, with some additional fields, that are primarily used for *Over The Air Provisioning* (OTA-Provisioning[15]).

Two fields are essential for OTA-Provisioning:

- *MIDlet-Jar-URL* specifies the location where the MIDlet-Suite (the .jar file) can be found on the server.
- *MIDlet-Jar-Size* declares the size of the referenced MIDlet-Suite in bytes. The value is used to:
  1. roughly verify the download candidate (by comparing the specified with the real file size).
  2. check whether the device has enough free memory for storing the application and if it supports applications with the given file size[16].
  3. ask the user whether s/he is willing to pay for the data-transfer of the given size.

Certification

Since the release of MIDP 2.0 it is possible to add a certificate to a .jar file to gain more rights for the MIDlet-Suite, e.g. access the storage system (the same that is used to store images, MP3 files, etc.). To achieve this, it must be able to authenticate the author, and the author must be classified as trustworthy. This is done by using asymmetric, cryptographic algorithms, based on the standard X.509, to sign the whole MIDlet-Suite [Sch07]. It is not sufficient that the author just signs the suite, because the user (i.e. his/her device) does not know whether the signature is trustworthy. Therefore the public key has to be signed by a *Certificate Authority* (CA), that guaranties the originality. Every device stores the public keys of the most important CAs and, thus, can check whether the MIDlet is trustworthy.

Because the signing process has to be done for nearly every mobile device separately, which obviously can become very expensive, this process is often neglected by most authors/publishers.

Programming Basics

A MIDlet has to extend `javax.microedition.midlet` and implement some basic methods:

Listing 3.1: Example for implementing a MIDlet

```
public class TestMIDlet extends MIDlet {
    protected void startApp() throws MIDletStateChangeException {
        ...
    }

    protected void pauseApp() {
        ...
```

[15]Find details here: `http://en.wikipedia.org/wiki/Over-the-air_programming`, accessed at 10.10.2007

[16]On some devices applications of various sizes can be installed, as long as there is enough free memory available. Others are more strict and specify a maximum jar-file size, e.g. 512KB for Nokia's 6230i [JPb].

```
    }

    protected void destroyApp(boolean unconditional) throws
        MIDletStateChangeException {
        ...
    }
}
```

These methods are essential, because they define a basic MIDlet-lifecycle as shown in figure 3.4. The three methods are called by the underlying operating system, to force a transition to the states *Paused*, *Active* and *Destroyed*.

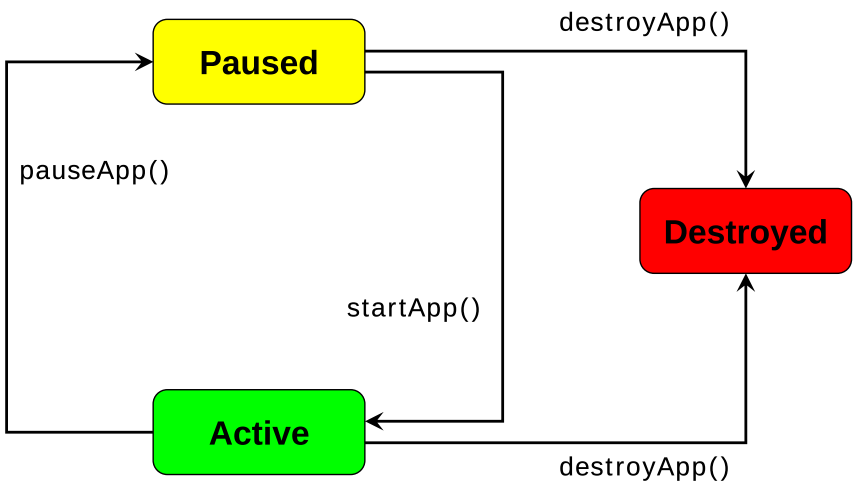


Figure 3.4: The MIDlet lifecycle.

- *Paused*: the MIDlet is initialized, but has no permission to output anything on the device's screen. Although background activities (e.g. calculations, network access) are not prohibited, they are not recommended, because another MIDlet may be active and competes for the occupied resources.
- *Active*: the MIDlet has access to the screen and receives user-input. Thus, this is the state, the application is usually executed in.
- *Destroyed*: the MIDlet stopped the normal program flow and released all resources. Depending on the application, the application's state may have been saved. Once reached, the state can not be changed any more and the program execution ends.

Access to Resources A *.jar* file may contain other resources than the used classes, which can be accessed by the MIDlet. Therefore, images, texts or other files may be included in the package and can be accessed by using the method `getResourceAsStream()` of `java.lang.Class`.

Because the classes of a package are already pre-verified to suppress manipulations, it is not possible to gain access to the *.class* files themselves (what makes *Reflection*[17] impossible).

Properties Two property types are accessible within a MIDlet (example code for accessing different properties can be found in listing 3.2):

- *System-Properties* define system specific attributes, like the user-defined language or available libraries (profile, configuration, or optional packages).
- *MIDlet-Properties* are configured in the *.jad* descriptor or the *Manifest* (find further information on page 17) and are either predefined MIDlet properties, or user-specified.

Listing 3.2: Example implementation that shows the access to different types of properties

```
public class TestMidlet extends MIDlet {
    ...
    private void getSomeProperties() {
        // System-Property (set by the host device)
        // e.g. ''MIDP-2.0''
        String midp = System.getProperty("microedition.profiles");

        // standardized MIDlet-Property
        // e.g. ''/img/icons/midlet.png''
        String icon = this.getAppProperty("MIDlet-Icon");

        // a custom property (specified in MANIFEST.MF, and the .jad
            descriptor)
        // e.g. ''192.168.0.1''
        String host = this.getAppProperty("Server-Address");
    }
    ...
    // implement the three abstract methods, beside other logic here
}
```

High- and Low-Level GUI The two approaches of programming a user interface lead to the differentiation of the corresponding APIs into a Low- and High-Level API or GUI, which is illustrated in figure 3.5.

Where the High-Level GUI is used to present the user a unique look on different devices, the Low-Level GUI can be seen as an empty sheet of paper (the `Canvas`) for drawing lines, curves, points or texts on. The High-Level API consists – similar to HTML – of elements like lists, text-boxes, alerts and forms (see figure 3.5), where the latter themselves are built upon elements like text fields, images or choice groups (similar to HTML's checkbox- or radiobutton-group). The concept of user input for primitive elements like a text-box or choice-group is obvious – via joystick and keypad – and transparent for developers. Because it is often necessary to react on user input, it is possible to register an `ItemStateListener`, which is similar to an *Observer* of the *Observer Pattern*. Additionally, each `Displayable`, thus all high-level elements and the low-level `Canvas`, can be assigned several `Commands` with associated actions. These can be used

[17] `http://en.wikipedia.org/wiki/Reflection_(computer_science)`, accessed at 15.11.2007

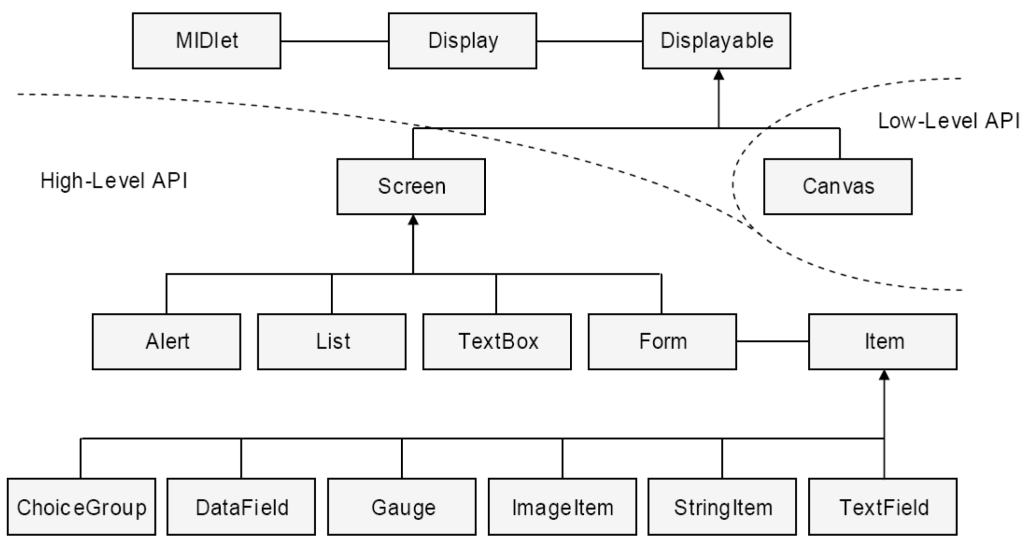


Figure 3.5: Overview of *JavaME*'s Low- and High-Level GUI.

to create menu entries, for general (e.g. "next" or "exit"), and specific (e.g. "add new item") commands.

The main usage of the Low-Level API can be found in mobile games, where developers need full control of the screen and the device's keys. Another reason for choosing the Low-Level API is when the user interface should look the same on every device. This is not possible using the High-Level GUI, because the design of high-level elements is not specified, and vendors tend to implement them differently e.g. to support corporate identity. For this special use case, the open source project *J2ME Polish* (find further information on page 24) has been founded, which lets developers use a special High-Level GUI, that can additionally be configured by using a subset of *Cascading Style Sheets* (CSS)[18]. As J2ME-Polish allows to specify a different CSS file for each device, the application can be customized to look the same on all handsets. The same way, this capability can also be used to define different look&feels for different devices or vendors.

Record Management System Because it is sometimes useful to store data on a physical, non-transient medium, MIDP specifies the *Record Store Management System* (RMS) that is accessible via a device-independent interface. It is a record oriented database with a unique name and fields of variable length and type. When a new record is added to the database, it is assigned a unique identifier, which has the same functionality as a primary key in relational databases. Records can be accessed directly (via its unique ID) or sequentially (one by one), where classes for sorting or filtering (that have to be implemented individually) may help to navigate. However, a RMS can not compete with relational databases like *MySQL* or *Oracle*.

All MIDlets of a MIDlet-Suite have the same access rights – read and write – to record stores that have been created by one if its MIDlets. Applications of other MIDlet-Suites are, per default, not able to access the RMS, although such access rights can be defined on creation.

[18]CSS has its roots in web-design, where it is used to specify a certain user-interface design (colours, text-sizes, . . .).

IDEs

Integrated Development Environments (IDEs) are used for implementation efforts in various programming languages because they help writing error-less code in short time. They support developers in writing, compiling, testing, debugging and executing code and sometimes they transparently transfer the executables to a target host system.

In principle there is no difference between IDEs for developing desktop applications and those for developing *JavaME* applications. But, if you take a closer look, you will see that the whole software development process (as described in subsection 3.1.2) is quite different to those of desktop or server applications, and an IDE must support every phase of it.

Actually, there are two well established IDEs that support the development for *JavaME*:

- *Eclipse*[19] with the *EclipseME*[20] plug-in lets developers write, compile, obfuscate, pre-verify, package and emulate their applications. New versions of the plug-in allow the use of pre-processing instructions to customize the application based on certain criteria (e.g. the screen size).
- *Netbeans*[21] with the *Mobility Pack*[22] supports all the functionalities of EclipseME, but integrates it seamlessly into the IDE (both, the IDE and the plug-in are developed by Sun). Further advantages are for example the *Visual Design Editor*, for drag&drop GUI-creation or the integrated localization support.

To round up this overview, the list below shows tools that are IDE-independent, although they provide interfaces to be used in major IDEs like Eclipse or Netbeans:

- *Antenna*[23] specifies *Ant*[24] tasks for pre-processing, building, obfuscating, pre-verifying, packaging, signing and running MIDlets.
- *J2ME Polish*[25] is based on Antenna, but enhances the building process with the possibility of resource assembly based on vendors, devices or device characteristics.
- *Maven 2*[26] with the *j2me-maven-plugin*[27] (which is also based on Antenna) allows the integration within Maven2 project hierarchies and the use of other Maven2 plug-ins (e.g. the *maven-assembly-plugin*[28]).

Emulators/Emulation

Sun provides with the *Wireless Tool Kit* (WTK) a few emulators to run MIDlets on a PC — they present important classes of cell phones (colour/black-white display, different screen sizes,

[19]`http://www.eclipse.org/downloads/`, accessed at 10.10.2007
[20]`http://www.eclipseme.org/`, accessed at 10.10.2007
[21]`http://www.netbeans.org/products/ide/` (accessed at 10.10.2007), please find further details in [Hla06]
[22]`http://www.netbeans.org/products/mobility/`, accessed at 10.10.2007
[23]`http://antenna.sourceforge.net/`, accessed at 10.10.2007
[24]`http://ant.apache.org/`, accessed at 10.10.2007
[25]Find further information on page 24.
[26]`http://maven.apache.org/`, accessed at 10.10.2007
[27]`http://pyx4me.com/pyx4me-maven-plugins/j2me-maven-plugin/introduction.html`, accessed at 10.10.2007
[28]`http://maven.apache.org/plugins/maven-assembly-plugin/`, accessed at 10.10.2007

etc.). Nearly every vendor (Nokia, Motorola, Sony-Ericsson, ...) does the same, and provides emulators that simulates their handsets on PC, not only in visual, but often also in behavioural ways. These emulators are essential for the developing process, because they give a first impression of the application without the tedious process of creating a MIDlet-Suite, transferring and installing it on a real mobile phone. Beside these quick-checks, emulators are very useful for debugging or simulations, where more than one cell phone is involved. They are usually very intelligent and allow even to assign different telephone numbers to different instances and send SMS messages from one to another. Some of them also simulate the processing power of the emulated mobile, where calculations take pretty much the same time as they would do on a real phone.

It is often useful to use an emulator for presenting an application to customers, colleagues or testers. Because those people often don't want to bug around with the installation of software they use presumably once, online-emulators[29] are ideal for this purpose. They are either integrated into a web page using *Java-Script* or *Java* and do nearly the same as other emulators do, but with hardly any software requirements. For some emulators it is sufficient to enable Java-Script, if this is not already done, others need an installed *Java Runtime Environment* which is likewise installed on most PCs. A few rules of thumb can be kept in mind, when using (online-) emulators:

- They may look different and behave differently than other emulators or real devices.
- Online-emulators based on Java-Script usually do not support MIDP 2.0.
- Usually resources like SMS, Bluetooth or access to storage devices are not supported and can't be simulated.
- It may be, that users need to install a *Java Runtime Environment* or activate Java-Script in their browser.

Nevertheless, developers must keep in mind, that emulators can't replace real devices — especially for testing and customizing purposes.

Installation

There exist 3 general ways to install the application on your mobile:

1. Surf to a Web- or WAP-page and download the application (usually by clicking an URL that points to a .jad file)
2. Trigger (e.g. your operator's call-centre initiates) a WAP-push which has a similar effect as in 1, except that the user does not have to manually enter a specific URL — the device does it for him/her.
3. Download the application to your PC and transfer it to your mobile via a data-cable or Bluetooth (use *OBEX push*[30]).

[29]For example the *mpowerplayer* (`http://mpowerplayer.com/`, accessed at 10.10.2007).
[30]`http://en.wikipedia.org/wiki/OBEX`, accessed at 15.11.2007

J2ME Polish

The most confusing part of this framework is its name, that suggests it has to do anything with Poland. Robert Virkus, the inventor and main developer of the project, clarifies on the project's website:

> "J2ME Polish should mean "turtle wax", at least that's what slashdot reader tod suggests[31]. The name can be confusing really. It has nothing to do with Poland or the language spoken there, rather it should imply that you can polish up your applications with J2ME Polish." [JPa]

The advantages of the project are not only limited to APIs:

- *Resource Assembling*: specify individual resources for different target devices. It's also possible to specify needed capabilities such as "supports MIDP2.0 and screen-size is 176+x208+".
- *Building J2ME applications*: specify target devices, individual .jar files will be created for. The build process supports the phases: pre-processing, resource-assembling, compilation, obfuscation, pre-verification, packaging and .jad file creation.
- The *device database* plays an important role in the above two points. It is used to select devices, based on capabilities specified in the resource assembling, or in the pre-processing phase, to generate individual code.
- The framework supports *CSS like GUI specifications*, which can additionally be enriched with *bitmap-fonts*, that make the application look the same on different handsets. Combined with the resource assembling these are the most powerful advantages of the J2ME-Polish project.
- A speed-optimized *game engine* allows the execution of MIDP 2.0 applications even on MIDP 1.0 devices.
- The *logging framework* makes `System.out.println()` calls visible, even on real devices where they usually are suppressed. Like for other logging-frameworks, the log-level ("debug", "warn", "error") can be chosen.
- Stand-Alone tools, like the *Font-Editor* let developers do their jobs more easily.

Using the framework can be done in two ways: either by using high-level GUI elements of the provided J2ME APIs and compiling it with the J2ME-Polish script, which replaces the elements with *polished* ones. Or, by directly using J2ME-Polish components, what creates apparent dependencies to the framework. The advantage of the first approach is, that developers can decide not to use the framework and stay with the used API. This could be very useful for compatibility reasons, although the link to the produced application is not clear at first sight.

J2ME-Polish is an open source project, but is available in two license flavours:

- Everybody can freely use features of the whole project (find an overview above), but has to publish the resulting code under the *GNU General Public License* (GPL)[32].

[31] `http://developers.slashdot.org/comments.pl?sid=116170&cid=9831734`, accessed at 10.10.2007

[32] `http://www.gnu.org/licenses/gpl-3.0.html`, accessed at 10.10.2007

- For commercial publications, where the code needs to be confident, you can buy a commercial license — this option also includes extended support, depending on the bought package.

Although J2ME Polish is an open source project and you are free to extend or modify it where needed, it must be said that it is not an easy task to do so. The code is hard to read, and thus, hard to change or extend, what makes it nearly impossible to adapt it to the developers' needs.

A test for introducing the J2ME-Polish elements to the game template 1 of the *mGBL* project revealed that the required GUI design could not be realized with the given API. Nor was it possible to modify the components to fit our needs, because features and parameters are spread over several classes and it's not clear where, in the code, modifications are necessary. Concretely, a text box of fixed height should have been implemented, which is scrollable when the content does not fit into the box. The API provides a component with similar functionality – a scrollable box that can contain various other elements – but the height can't be limited, unless it fits into the screen. It seems, that the aim of this box is to be as small as possible, but enlarges when needed.

Summary

JavaME might look complicated at the first sight, but has some unique advantages, and therefore makes the race – at least for the application implemented within the *mGBL* project – against other competitors. Reasons, that speak for *JavaME*:

- It is widespread. There is hardly any mobile phone or smart phone that is not able to run a *JavaME* application.
- It is free. Even the most popular development tools are freely available[33].
- *JavaME* has a steep learning curve, especially for developers who are already familiar with Java.
- A lot of API's designed for Java are available in a minimal version that is customized for the limited resources available on mobile devices[34].
- It is customizable. Using the Low-Level-GUI-API, it is possible to make a user interface look whatever you like.

3.2.2 Symbian OS

Symbian OS is a proprietary operating system designed for mobile devices, produced by *Symbian Ltd.* The company exists since 1998, was founded as a cooperation of different mobile phone producers, and the company Psion [Ess06], and is now owned by the companies Nokia (47.9%), Ericsson (15.6%), Sony Ericsson (13.1%), Panasonic (10.5%), Siemens AG (8.4%) and

[33]The IDE's *Eclipse* (`http://www.eclipse.org/`) and *Netbeans* (`http://www.netbeans.org`) are free of charge as well as the needed plug-ins (that is, the *Mobility Pack* for Netbeans, and *EclipseME* for Eclipse).

[34]For example the *kXML* (`http://kxml.org`) XML-reader and -writer API.

Samsung (4.5%) [Ess06, Wiki]. Seeing these share rates, it is clear that Nokia is currently the producer with the most Symbian based mobile phones on the market [Ess06].

Symbian defined different version of its operating system, and called them *Series*, where each Series targets different purposes. The *Series 60* (codename "Pearl"), the most spread one, is trimmed to meet the requirements of smart phones, which have a normal cell phone keyboard, and can be operated one-handed. Phones based on the *Series 80* (codename "Crystal"), usually have good organizer functionality, a QWERTY[35] keyboard, a large display, and are equipped with a fully functional e-mail client and web-browser. *Series 90* enabled devices stand out with their large displays and touch screen functionality. Another important series is the *UIQ* (codename "Quartz"), which is designed for devices that are a combination of cell phone and PDA. Again, a large, touchable display and a stylus are characteristic for those devices, which are especially used for professional purposes [Ess06, Wiki]. A prominent exemplar that makes use of the UIQ is Sony-Ericcson's *P1i*[36], which has a built-in two-megapixels-camera, an integrated business card scanner, a web-browser and an e-mail client, to name just a few features.

Developing on Symbian OS

The operating system is not open source, although parts of it are publicly available, to allow developers produce code for the platform. This enables the implementation of development tools for different programming languages, like Python, Visual Basic, Perl and *JavaME* [Wiki]. However, the favourite programming language, still, is C++, because it allows the development of machine-oriented, thus faster, applications, with hardly any restrictions.

Symbian OS C++

The term "Symbian OS C++" refers to the C++ dialect, that is domain specific to the frameworks used to build the operating system itself, and the applications that run on it [HS07].

Symbian want its C++ dialect to be known as different to other C++ dialects for two reasons. First, it is a fact that C++ tends to differ from platform to platform. Second, as the combination of C++ and Symbian has an over 20 years history, it can be considered perfectly adopted to the system's needs.

3.2.3 .Net Compact

> "The .NET Compact Framework is a programming interface and runtime library created as a combination of two Microsoft technologies: Windows CE, an operating system for mobile and embedded 'smart devices', and .NET, Microsoft's reinvention of its programming interfaces and its developer tools." [YD04]

This chapter summarizes information about the host operating system on Windows enabled devices, *Windows CE*, presents basic principles of the *.Net Framework*, and then moves on to its little brother the *.Net Compact Framework*. The latter, of course, is the real topic of this chapter, however the other sections may help to understand how it works.

[35] QWERTY specifies a layout of keys, one is familiar with from typewriters or PC-keyboards (`http://en.wikipedia.org/wiki/Qwerty`).

[36] `http://www.sonyericsson.com/cws/products/mobilephones/overview/p1i?cc=en&lc=UK`

Windows CE

The first version of Windows CE was released in 1996. Although it was built with the Win32 API as its core, it shares hardly any code with its heavier desktop pedant. Windows CE has been rewritten from scratch to fit the needs of hardware and software engineers, developing mobile and embedded devices [YD04].

On Windows CE, in contrast to other Windows operating systems, the Win32 interface is still an option for developing applications, and even is in some cases the only possibility to accomplish some certain tasks.

The .Net Framework

The .Net framework is some kind of an umbrella that incorporates interfaces and tools for developers' needs in different divisions and combines them to a larger strategy. Best known are these three application classes: Windows Forms, Web Forms and XML web services.

Some of the main design goals of the framework are [Wikh]:

- Interoperability — because the interaction of new and old applications is often required, the .Net framework provides access to programs that are executed outside the .Net environment.

- Language independence — Microsoft introduced a common type system, where all possible data types are defined. Therefore .Net allows development in different programming languages, and that, again, supports interoperability.

- Security — the .Net framework allows the execution of code at different trust levels, without the need of a sandbox.

To allow the above mentioned language independence, a special architecture has been developed, where one specific programming language is compiled into an independent representation. The *Common Language Infrastructure* (CLI) takes care of this language independent code, and manages further compilation, which makes the code runnable on the target system (figure 3.6).

The .Net Compact Framework

The main goal of the .Net Compact Framework was to enable developers to use a known environment for developing applications for mobile devices. As you continue reading, you may realise, that the correlation between the .Net Framework, and the .Net Compact Framework is similar to the one of Java and *JavaME*.

The .Net Compact Framework was conceived as a subset of the .Net Framework in the full knowledge, that it can't be as extensive as its big brother. But it was designed to reach a maximum consistency between the two frameworks, concerning name spaces, classes, and types. While this makes the framework attractive to developers who are familiar with the .Net Framework, a lot of work has been invested into performance tuning, which should attract former C and C++ developers, too. The Compact Framework team, therefore specifies these design goals [YD04]:

- Build on the benefits of the .Net Framework

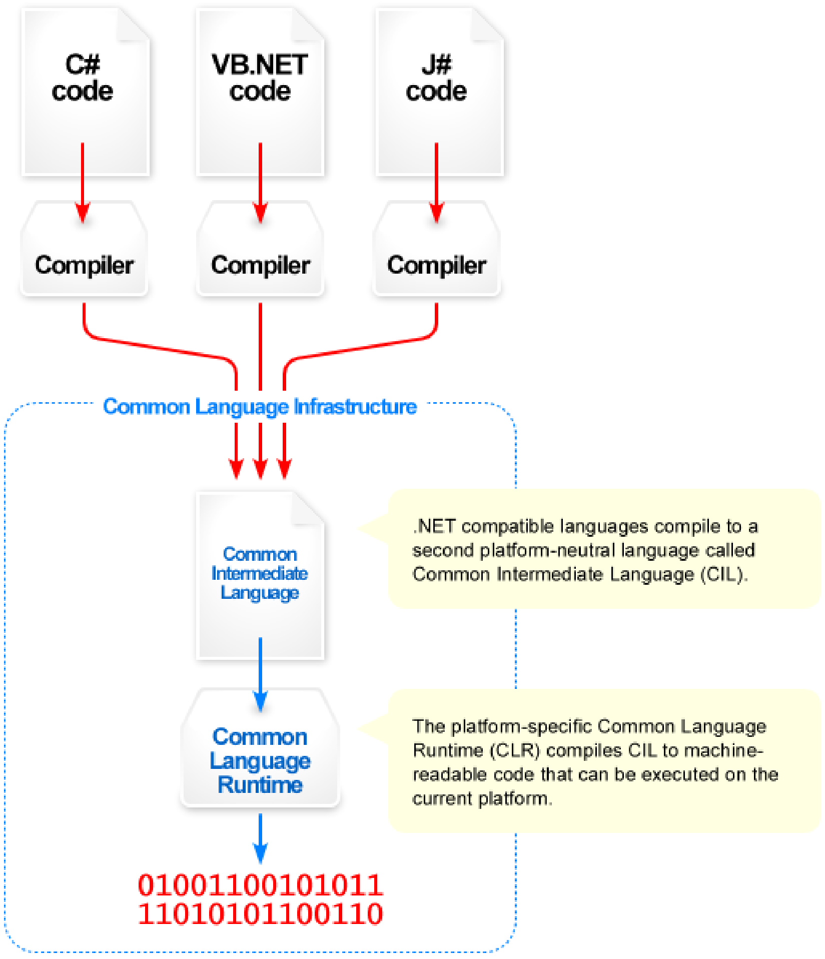

C# code
VB.NET code
J# code
Compiler
Compiler
Compiler
Common Language Infrastructure
Common Intermediate Language
.NET compatible languages compile to a second platform-neutral language called Common Intermediate Language (CIL).
Common Language Runtime
The platform-specific Common Language Runtime (CLR) compiles CIL to machine-readable code that can be executed on the current platform.
01001100101011
11010101100110

Figure 3.6: Overview of .Net's Common Language Infrastructure (CLI). (source: `http://en.wikipedia.org/wiki/Image:Overview_of_the_Common_Language_Infrastructure.png`)

- Maintain consistency with the desktop — In order to scale down the footprint of the framework to 2MB, the team eliminated many of the desktop framework classes, trimmed the remaining class members, and reduced properties, methods and events.
- Run well on mobile and embedded devices — The reduced size of 2MB fits very well to typical Windows CE powered mobile devices with 32-64MB RAM.
- Preserve the look&feel of platforms — To a user, a Compact Framework application is visually indistinguishable from non Framework applications.

Programmers, who are familiar with the .Net Framework, are high qualified candidates to get productive with the Compact Framework. Nevertheless, the significantly reduced feature set makes it essential to allow time to get comfortable with it, since many favourite classes and class members seem to be missing.

Xbox 360

A version of the .Net compact framework is also available for Microsoft's game console *Xbox 360*[37]. This framework is similar to the .Net compact framework, which is optimized for mobile devices, also a subset of the .Net framework, but is optimized for the Xbox 360. [Wikh]

3.2.4 Flash Lite

Adobe's *Flash Lite* is a lightweight version of the Adobe Flash Player, optimized for mobile phones, and other devices with limited resources [Wika].

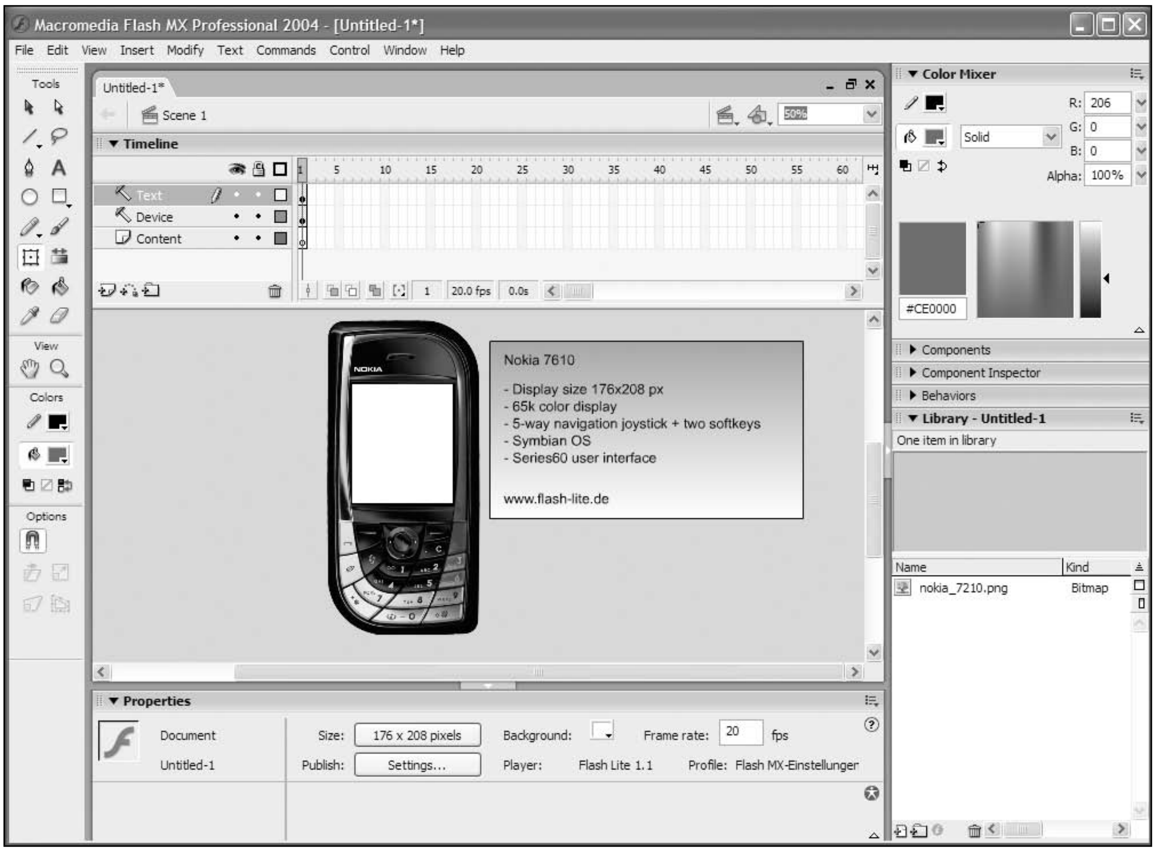
Flash for mobile devices comes in those three flavours [LJD07]:

- *Flash for Pocket PC* — a version for a PDA, which supports the whole functionality of the Flash 6 and Flash 7 player (including XML, UI components and data loading).
- *Flash Lite* — a profile, especially developed for mobile phones.
- *FlashCast* — a streaming web technology that utilizes the Flash Lite player.

Flash Lite 1.0 and 1.1 are based on the Flash 4 player, include Action Script, but do not support all of the functionality of the desktop profile. It, of course, supports vectors, bitmap graphics MP3 sound and a scripting language and others. However, Flash 2.0 is based on Flash 7, which supports Action Script 2, and should therefore be more familiar to developers who have experiences with Flash desktop applications.

Good news for all Flash developers is, that the *Flash MX 2004 IDE* provides you with everything you need to design, produce and test your Flash Lite application. You can even download different templates for different target devices, that reflect the particular restrictions like the display size, and provide an authentic shape of the mobile phone for the emulation of applications (figure 3.7).

[37]`http://www.xbox.com`



Nokia 7610

- Display size 176x208 px
- 65k color display
- 5-way navigation joystick + two softkeys
- Symbian OS
- Series60 user interface

www.flash-lite.de

Document: Untitled-1
Size: 176 x 208 pixels
Background:
Frame rate: 20 fps
Publish: Settings...
Player: Flash Lite 1.1
Profile: Flash MX-Einstellungen

R: 206
G: 0
B: 0
Alpha: 100%
Solid
#CE0000

Name	Kind
nokia_7210.png	Bitmap

Figure 3.7: Flash MX IDE with the Nokia 7610 Template [LJD07].

Disadvantages

Flash Lite applications are not capable of communication via Bluetooth, infra-red, or with a mobile phone's built-in camera [Wika].

The license of the developer suite as well as the Flash Lite player are not very expensive, but also not for free [LJD07].

Advantages

It is quite easy to produce a fancy, graphically affine application, even for people that have few or no programming skills. Therefore, the time-to-market is reduced to a minimum in contrast to competitive products, because the most challenging tasks are quite often design, implementation and especially the adoption of the graphical user interface to different devices.

Chapter 4

Mobile Game-Based Learning

As already said at the beginning of this work, *Mobile Game-Based Learning* is an EU project of the 6^{th} framework programme, and was initiated to investigate possibilities to support learning on mobile devices. Therefore, three prototypically game templates, a game management platform and appropriate authoring tools should be specified and implemented. Each of these components will be explained shortly, in the following sections, to give an impression of the whole architecture, concluding with the focus of this work.

4.1 Web-Platform

Figure 4.1 shows a schematic representation of the web platform. Some of the components are integral parts of the platform, whereas others are developed separately. The user management, as well as the OR-mapper (not shown in figure 4.1), which is used to store object-oriented data into a relational database, are such integral parts.

The schematic diagram, figure 4.1, is not complete, but shows all important parts necessary to understand the web platform and the overall architecture.

User Management is spanned through the whole application, because only registered authors are allowed to create or change content. The system handles different types of users, which inherit different access rights, because learners also have to have access to some parts of the platform to download and manage their games or change personal settings, like user-name and password.

Game Management inherits the creation of games by authorized users, as well as the management of who and when people are allowed to download and use a game. It is possible to assign single users, or whole groups to a particular game. Additionally, a start and end date can be specified, within the before mentioned users are allowed to download the game, and let it communicate with the server.

The Game Selection Tool helps authors to decide for a particular type of game, by specifying their requirements and constraints. Currently, this tool is stand-alone, in the way that it has no direct connection to any game authoring tool. Instead it serves as advisor, that can be used to pick a game outside of the *mGBL* context. When, some time, more game templates are available, the tool can be seamlessly integrated into the game creation process.

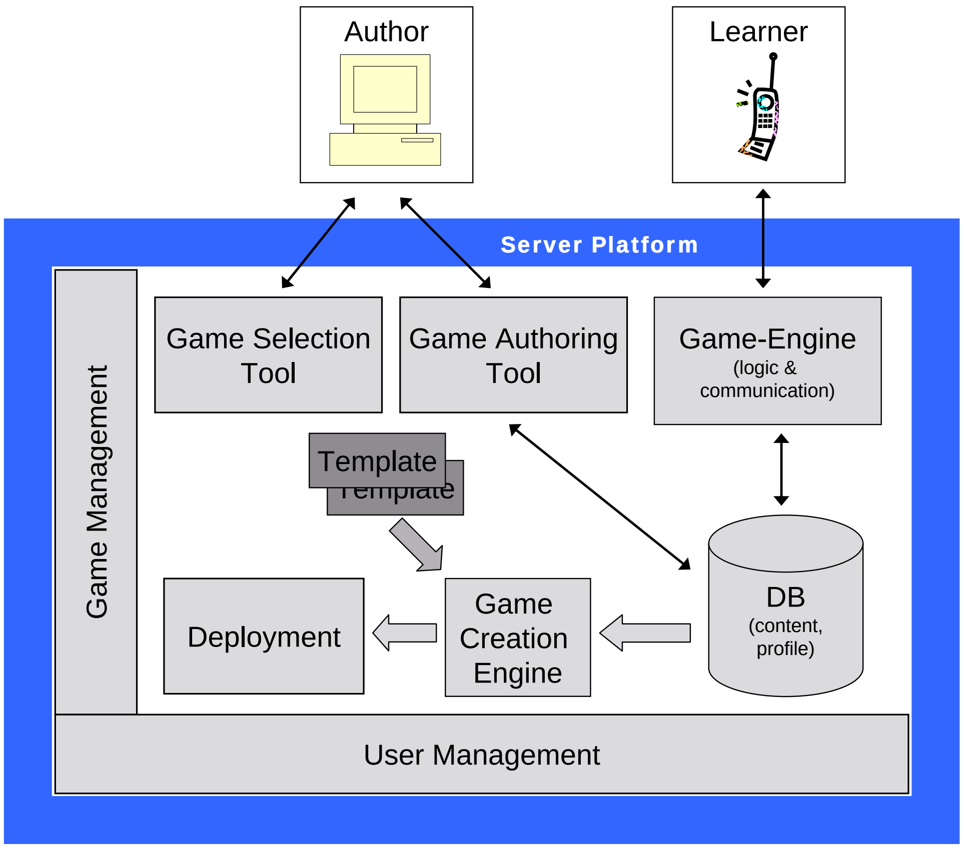


Figure 4.1: Overview of the web platform, showing the most important components

The Game Authoring Tool is separately developed for each game template, and tailored to each specific need. Therefore, it should rather be called "authoring tools"[1]. One such tool is directly and transparently connected to the game template it is implemented for. Therefore the introduction of a new game template requires its connection with the appropriate authoring tool.

4.2 Templates

The *mGBL* project demands at least three different game template prototypes as one of its outcomes.

During the authoring process, the author selects one of the implemented templates and creates content explicitly for that template. Both are combined by the system, and are compiled to one playable game (or game component, see subsection 5.1.1). Actually, the web platform produces not just a game, but several instances of the same game, that are (potentially) tailored to different devices (see section 4.3 for details).

4.3 Distribution

The deployment component, sketched in figure 4.1, is responsible for creating different versions of one game, which are tailored to different target devices. This may be necessary for some game templates, because they are designed for one, or a few, particular handsets, and need to be "transformed" to fit other devices[2].

As described in section 4.1, the so created different versions of one game can be downloaded via the web platform, by authorized users.

4.4 Communication

For a better readability, a so called *Asynchronous Communication Component* has been left out in figure 4.1. Therefore, for SMS-based, thus asynchronous connections, a handset does not directly communicate with the Game Engine, but with the Asynchronous Communication Component. It queues incoming requests, and schedules their processing by the Game Engine. The other way around, it queues outgoing messages and reliably sends them to the target handsets.

4.5 Roles & Tasks

Talking about a game template or the web-platform, the term user arises quite often. Keep in mind, that this term can be used when talking about an author, a player, or a validator.

[1]Find the design of the authoring interface for game template 1 in appendix C.

[2]It, indeed, would enrich the possibilities of *Ahead of the Game*, because such a tool would enable us to adopt the game to many different mobile phones, by just creating a few, slightly different setting files (including for example graphics in different resolutions). However, this step has not been taken so far.

Simply speaking, a user is someone who uses the software. Following, we define three roles a user can take.

4.5.1 Roles

A game author can be a teacher or a student. In principle everybody can author a game, but if games are used in learning groups, they need to be validated by a game validator, who is usually the teacher. Games are usually played by students (game player), nevertheless teachers may want to verify their game by playing it themselves. However, it is unclear, yet, if an author can play his/her games via an emulator or participates the competition like other learners (see table 4.2). Table 4.1 gives an overview of the most important roles[3].

Table 4.1: Different user-roles.

	Author	Validator	Player
Teacher	x	x	
Student	x		x

4.5.2 Tasks

As shown in table 4.2, the tasks of a teacher are authoring and validating a new game, as well as administering learner-groups. The major task of a student is playing a learning game. The student/player can also view and alter his/her personal profile and inspect his/her learning progress.

Table 4.2: Tasks of the most important roles.

	Teacher	Student
Authoring	x	x
Validating	x	
Playing a game	o[4]	x
Creating and maintaining one's own profile	x	x
Viewing one's learning progress	x	x
Administering a learning group	x	

4.5.3 Server-side & Client-side Tasks

All games are authored on the server. For each game type a specific authoring interface is available. The authored game must be validated by an authorized validator, who is typically

[3]Other roles may be useful for the web-platform (e.g. the role "Administrator"), but are neglected here.

[4]Teachers may want to verify their games by playing them themselves, but it is unclear, now, how this will be supported.

the teacher. Administering of learning groups also takes place at the server. Moreover, the students/learners create, view and maintain their profiles, as well as view their overall learning progress on the server. Table 4.3 summarizes all important activities, while focusing on the two platforms, unlike table 4.2, which shows the same tasks in contrast of different users.

The client application is mainly a framework for playing the authored game. However, the user's full profile, containing (beside other data) scores of all played games, is stored and maintained on the server.

Table 4.3: Tasks in contrast of the two platforms.

	Server	Client
Authoring	x	
Validating	x	
Playing a game		x
Creating and maintaining one's own profile	x	
Viewing one's learning progress	x	x
Administering a learning group	x	

4.6 Focus

This work focuses on the specification and implementation of one game template — *Ahead of the Game*. Although details of the web-platform will be mentioned in the next chapters, it is just for the reader's better understanding of the whole picture.

In the next chapters, the specification of the two components of *Ahead of the Game* is shown, including their game flow and interaction with each other. The specification concludes with the game's scoring concept, controls, and non-features, that are features that might be associated with the application, but are not going to be implemented for certain reasons[5].

The following chapter (chapter 6), explains how the specification is implemented. It starts with the used technology and a package overview, before explaining the underlying software design. Based on the data model, the basic functionality and potential of the application is explained, which is rounded off by an overview of important settings to configure the application. The implementation chapter concludes with other interesting aspects, like the server communication, file structures, user profiles, and possible extension points for future implementations.

[5]See section 5.5 for more details.

Chapter 5

Template Specification

5.1 Overview

This chapter gives you an overview of the game template 1 *Ahead of the Game*. First I will explain in some simple statements how the game-flow works, and continue with more details.

5.1.1 Components

There are two major components — the quiz (*Fastest First!*) and the simulation component (*Crisis!*). While the first is used for testing the user's knowledge, the latter is used to put him/her into a virtual emergency scenario (e.g. a car crash for e-health, or an imminent inflation for e-commerce). The quiz component lets the user know if s/he has done right and how much points s/he has gained, for each question, where in contrary the simulation – like in real life – does not give immediate feedback. Although, if the user asks for, s/he gets detailed feedback at game end, via the web-platform. The major idea behind those two components is that the one is used to teach the user knowledge that s/he has to apply at the simulation phase.

5.1.2 Game-Flow

Figure 5.1 gives an overview of how the game is composed of the two components, quiz and simulation. The quiz component is played first, where the player gets questions in sequential order, concluded by an upload of the scores to the web-platform. The simulation has a more cyclic nature, where the user returns to the same, but changed (e.g. deteriorated), situation. Before the score of this component gets transferred to the server, too, the user has to assess his/her own doing.

Additionally, figure 5.2 repeats this presentation in a more visible way, enriched with more details according both components. In this schematic representation, the sequential order of quiz questions and simulation situations is more obvious.

The two game components can be authored/played independently, which results in the following possibilities:

- Only quiz
- Only simulation
- First quiz, then simulation

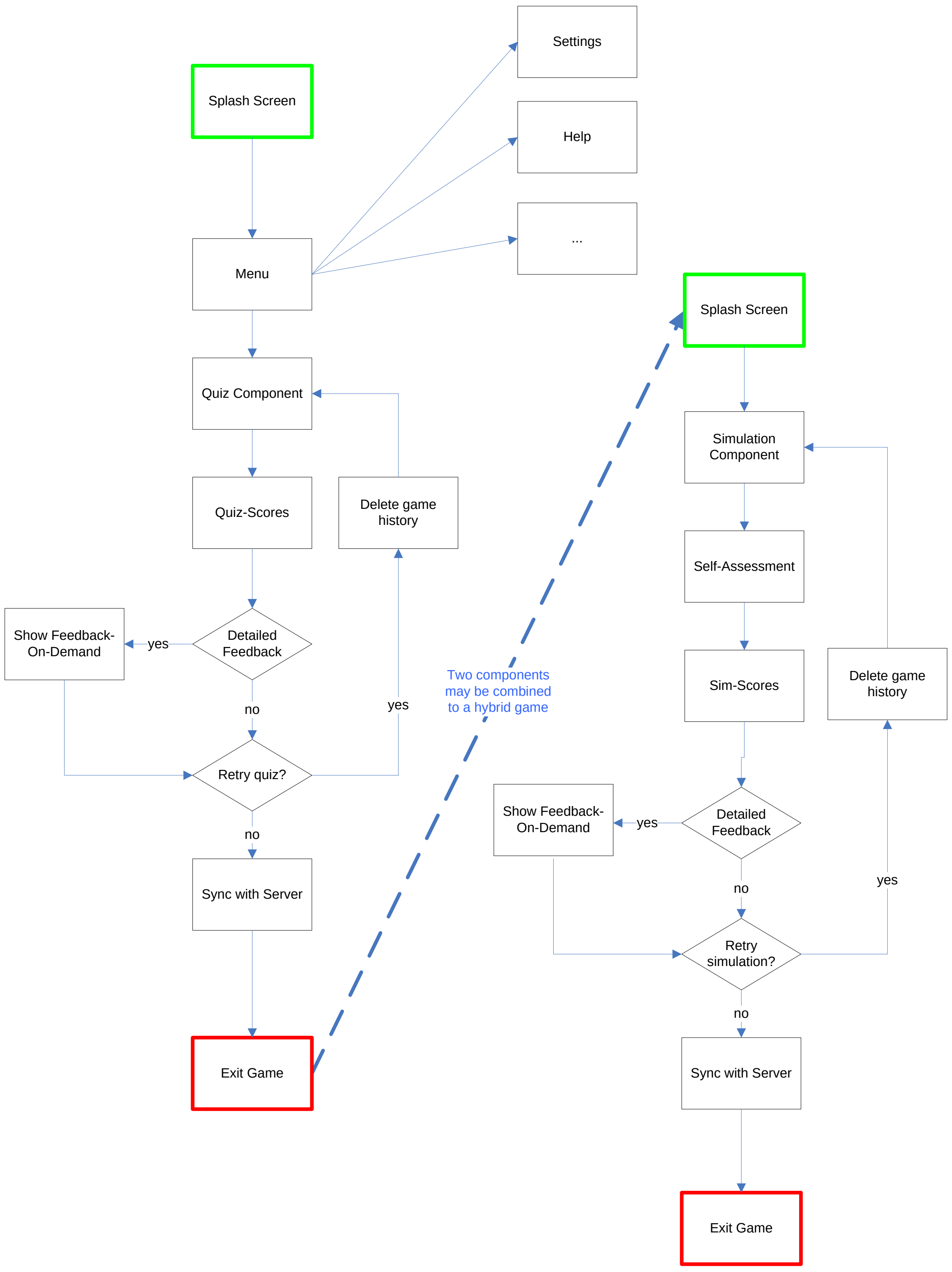
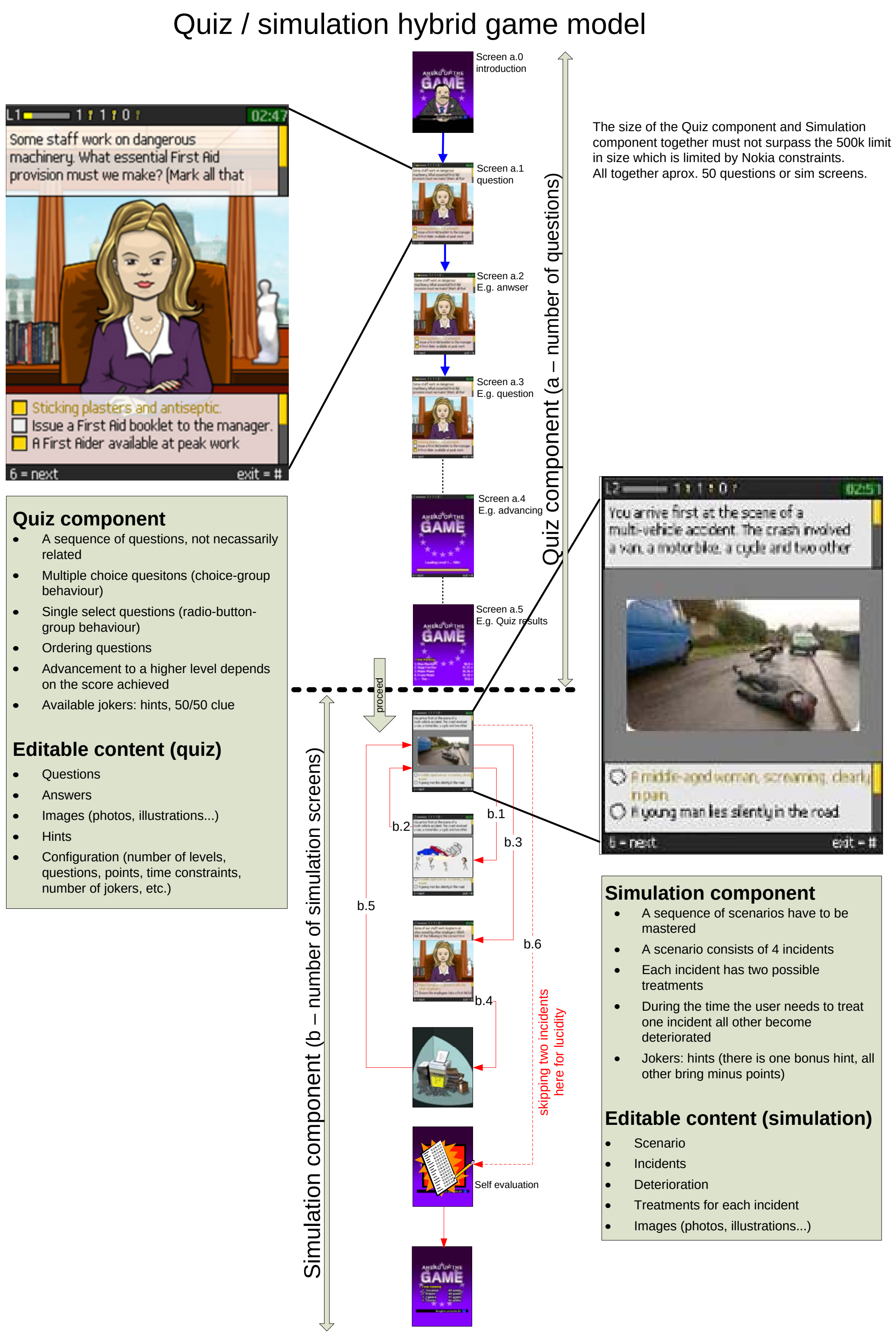


Figure 5.1: Game flow overview


Quiz / simulation hybrid game model
Screen a.0 introduction
Screen a.1 question
Screen a.2 E.g. anwser
Screen a.3 E.g. question
Screen a.4 E.g. advancing
Screen a.5 E.g. Quiz results
Quiz component (a – number of questions)
proceed
b.1
b.2
b.3
b.4
b.5
b.6
skipping two incidents here for lucidity
Self evaluation
Simulation component (b – number of simulation screens)
L1 02:47
6 = next
exit = #
L2 02:51

Some staff work on dangerous machinery. What essential First Aid provision must we make? (Mark all that

Sticking plasters and antiseptic.
Issue a First Aid booklet to the manager.
A First Aider available at peak work

The size of the Quiz component and Simulation component together must not surpass the 500k limit in size which is limited by Nokia constraints.
All together aprox. 50 questions or sim screens.

You arrive first at the scene of a multi-vehicle accident. The crash involved a van, a motorbike, a cycle and two other

A middle-aged woman, screaming, clearly in pain
A young man lies silently in the road

Quiz component

- A sequence of questions, not necassarily related
- Multiple choice quesitons (choice-group behaviour)
- Single select questions (radio-button-group behaviour)
- Ordering questions
- Advancement to a higher level depends on the score achieved
- Available jokers: hints, 50/50 clue

Editable content (quiz)

- Questions
- Answers
- Images (photos, illustrations...)
- Hints
- Configuration (number of levels, questions, points, time constraints, number of jokers, etc.)

Simulation component

- A sequence of scenarios have to be mastered
- A scenario consists of 4 incidents
- Each incident has two possible treatments
- During the time the user needs to treat one incident all other become deteriorated
- Jokers: hints (there is one bonus hint, all other bring minus points)

Editable content (simulation)

- Scenario
- Incidents
- Deterioration
- Treatments for each incident
- Images (photos, illustrations...)

Figure 5.2: Overview of the whole game

5.1.3 Game Levels

Both, the quiz and the simulation component are split up – or may be split up – into several game-levels, for example, to separate different topic- or skill-levels. The technical realization, though, differs in that point from the specification to support more flexibility (see chapter 6 for more details).

5.1.4 Points

User-obtainable points can always be positive or negative and their amount can be freely chosen by the author (as long as they are whole-numbers). The exact implementation of calculating the points is specified in section 5.4.

5.1.5 Progression

In both components the user gets confronted with all CeLOs sequentially, as they have been authored. The author can choose whether to randomly rearrange the different answer-options of one question, or not (in this case they appear in the order as they were authored).

5.1.6 Game Saves

Every time a CeLO has been completely processed it will be saved to the mobile phone's internal memory. When the player exits the game, not having completed a question or a simulation, s/he has to do it again.

At game end the user has the chance to improve his/her score by playing another component of the same type. When the player is satisfied with his/her score the system manages the upload of relevant data to the server (achieved points, used jokers, etc.).

5.1.7 The Boss

At the beginning of the game, the user has to select his/her boss gender by visually pick one of two avatars (a female and a male boss). This avatar will be used to present to players the game challenges. It's imaginable that in a later stadium of the project a more extensive collection of avatars will be available, which underlines the playful character of the game template.

To build an indirect multi-player game, it was planned to position a nameplate on the boss' desk, where the name can change. Thus, at the beginning of each level, the current high-scores are retrieved from the server, to use the top-ranked competitor's name as the bosses name. While adding information of the current status of the ranking, it creates also competition, and incites the player to get a good score. However, this feature has not been implemented, yet, because of tight time constraints.

5.2 Component 1 — Quiz

5.2.1 Principle of game play

As shown in figure 5.3, the application can be paused (exit before game ended). That is checked first, at every game start. The game starts, either at the first game-level with the first question,

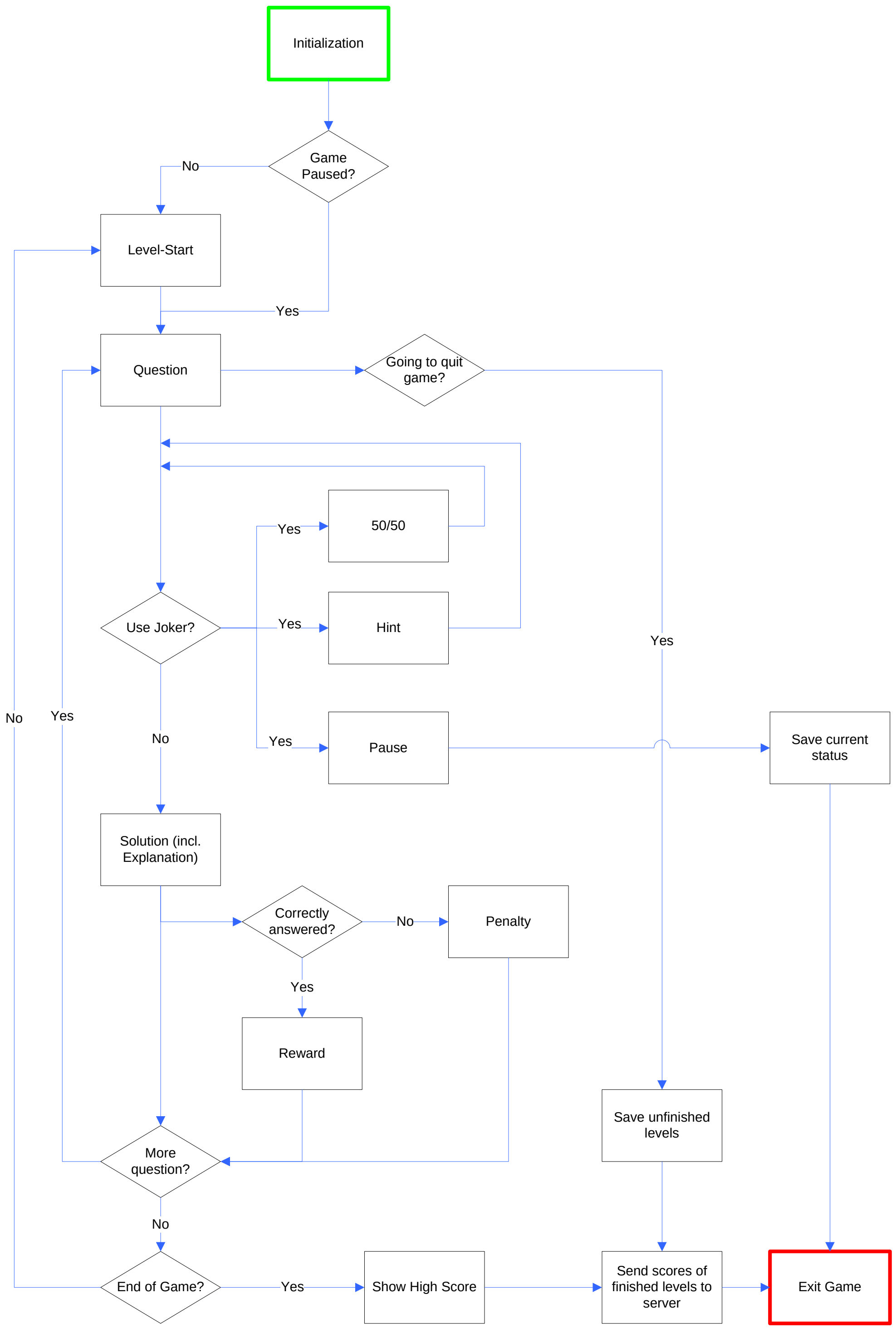


Figure 5.3: Gameplay of the quiz component

or at the saved position. While a question is presented to the user, s/he might use a joker, if allowed to. A joker is permitted, if the user has one or more left, and it is appropriate — for example a 50/50 joker is not appropriate for ordering-questions. When the user locked the answer-options, s/he gets feedback about their correctness, as well as an explanation of wrong ones. When the last question of the last game-level has been answered, the user's score is transferred to the server, which responds with a high-score-list that is shown to the user. On every game exit – at game end, or precipitate – the current game status is saved to the mobile phone's internal memory.

5.2.2 Jokers

In the quiz component the user is supported by three different types of jokers:

- 50/50 clue (can only be used with Single-Click questions)
- Hint (brings the user bonus or penalty points)
- Game pause

The author specifies how many of each joker-type the user can request during the game.

5.2.3 Feedback

The feedback is immediate — the user sees which answer-option was correct, which not and gets a short explanation why they are right or wrong. After the game, the player can obtain detailed feedback in respect of all the answer options.

5.2.4 Rewards/Penalties

The nature of the rewards and penalties makes it mandatory that the user can only earn one exemplar of each. Therefore the user earns one exemplar only at special occasions (when s/he gains a lot of points in relation to the maximum possible score).

Reward examples:

- Medal (e.g. “head of coffee machine”)
- A sarcophagus plant (for new office)
- Mow the boss' lawn
- Cellular phone to be reachable even at night

Penalty examples:

- shred financial accounting
- go shopping with the boss' wife (a lot of bags)
- act as clown at the next company party

Both, penalties and rewards have been intentionally designed with a bad touch, to keep the player's tension and motivation high.

5.3 Component 2 — Simulation

5.3.1 Principle of game play

As illustrated in figure 5.4, the user enters a simulation scenario, where s/he has to decide which of 4 critical cases to treat first (single click question). The prioritised case must then be "treated" by answering a second single click question. The default situation here is that there are only two treatment options per case, and the user will get no immediate feedback (see subsection 5.3.3). However, the author may decide to allocate more than 2 treatment options on a case by case basis.

After the user has decided and treated the first of the 4 cases, s/he returns to the remaining 3, which have meanwhile changed, e.g. deteriorated. It may even now happen that one of them "expires", which is best explained with a health scenario, where patients really could die. When we are not talking about people but about e.g. stock values, the case could be a stock that falls beyond a critical level.

The user has to take care of all 4 cases, prioritising at each stage. When all cases have been treated or have expired, the scenario ends and a new simulation may be requested.

5.3.2 Self-Evaluation

At the end of this component the user is asked to assess his/her progress (poor, fair, good, excellent), and gets a feedback whether this is true or not (e.g. "you are /not /almost/ right — your assessment: excellent, your score: good"). S/he then gets access to actual points scored. These steps are illustrated at the bottom of figure 5.4.

5.3.3 Feedback/Explanation on Demand

There is no instant feedback like in the quiz component (see subsection 5.2.3). The user rather has to cope with the simulation situation as it develops: the developments in the simulation represent the kind of feedback, provided by developments in real-life situations. However, the player can ask for detailed feedback at the end of the component, via the web-platform.

5.3.4 Jokers

On the one hand, no 50/50 clues are allowed (or necessary) because a simulation scenario consists of four cases, where none of the player choices (priorities, choice of treatment options) is absolutely wrong (it's more a question of what ought to be their priorities). On the other hand, some hints do make sense, and that's why one bonus-hint exists that brings bonus points for using it in the right situation, but all other hints bring minus points for using them.

5.3.5 Rewards and Penalties

Rewards and penalties work the same way as in the quiz component, please see subsection 5.2.4 for more details.

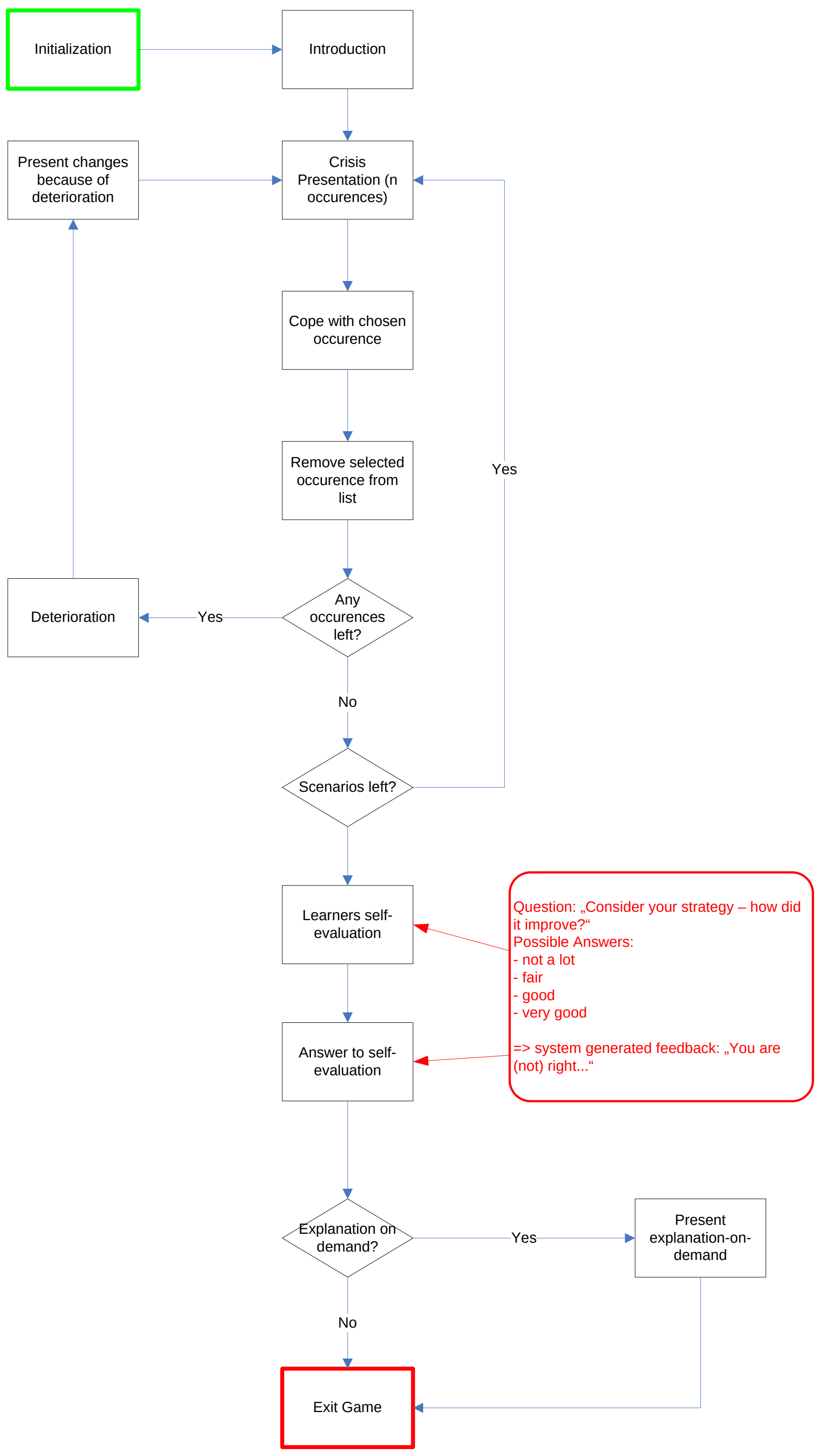

Initialization
Introduction
Present changes because of deterioration
Crisis Presentation (n occurences)
Cope with chosen occurence
Remove selected occurence from list
Yes
Deterioration
Yes
Any occurences left?
No
Scenarios left?
Learners self-evaluation
Question: „Consider your strategy – how did it improve?“
Possible Answers:
- not a lot
- fair
- good
- very good
=> system generated feedback: „You are (not) right...“
Answer to self-evaluation
Explanation on demand?
Yes
Present explanation-on-demand
No
Exit Game

Figure 5.4: Gameplay of the simulation component

5.3.6 Authoring Example

Please find an example of how a simulation component may be authored in table 5.1. For a better readability no cells for points or timing issues have been reserved.

Table 5.1: Simulation authoring example

Element / Time	t_1	t_2	t_3	t_4
Simulation description	You arrive first at a multiple car accident. What do you do first?	Who are you going to treat next?	Two people saved, who is next?	Last treatment…
Hint	Prioritise the life-threatening condition.	Prioritise the life-threatening condition.	Prioritise the life-threatening condition.	
Case 1	Man with head-wound	Man with bleeding head-wound	Man with bleeding head-wound	Man with bleeding head-wound
Priority	2	1	1	1
Case 2	Pregnant woman	Pregnant woman	Pregnant woman, screaming	Pregnant woman, screaming
Priority	4	4	3	3
Case 3	Boy sitting silent on the sidewalk	Boy sitting silent on the sidewalk	Boy lying silent on the sidewalk	Boy lying silent on the sidewalk
Priority	2	2	1	1
Case 4	Unconscious man	[The man died]		
Priority	1			

Simulation authoring example (continued)

Element / Time	t_1	t_2	t_3	t_4
Question for case 1	How do you treat the head-wound?	How do you treat the bleeding headwound?	How do you treat the bleeding headwound?	How do you treat the bleeding headwound?
Answer 1	Bandage	Compression bandage	Compression bandage	Compression bandage
Answer 2	Cleanse	Patch	Patch	Patch
Question for case 2	What's best for a pregnant woman?	What's best for a pregnant woman?	What's best for a pregnant woman?	What's best for a pregnant woman?
Answer 1	Lay down, feet up	Lay down, feet up	Lay down, feet up	Lay down, feet up
Answer 2	Sports	Sports	Sports	Sports
Question for case 3	A silent child - what do you do?	A silent child - what do you do?	A maybe unconscious child - what to do?	A maybe unconscious child - what to do?
Answer 1	Give him candy	Give him candy	Start a conversation	Start a conversation
Answer 2	Start a conversation	Start a conversation	Check the pulse	Check the pulse
Question for case 4	What's essential with unconscious ones?	[The man died]		
Answer 1	Free the airways			
Answer 2	Be very silent			

5.4 Scoring

This section describes the different scoring models according to the already described specifications (see sections 5.2 and 5.3).

Points can always be positive or negative, and their amount can be freely chosen by the author (as long as they are whole-numbers). The exact implementation of calculating the points depends on the specific implementation (quiz or simulation, see subsections 5.4.1 and 5.4.2).

5.4.1 Quiz

First of all, it should be clear that every question may bring plus or minus points, depending on the correctness of each answer-option, the used time and used jokers. The basic formula to calculate the points is:

$$P_{Question} = \sum_{Answer=1}^{4} (P_{Answer} + P_{Hint})$$

There is a switch in the game configuration that activates a time-based point calculation, which expands the formula to:

$$P_{Question} = \sum_{Answer=1}^{4} \left(P_{Answer} * \frac{t_{available} - t_{used}}{t_{available}} + P_{Hint} \right)$$

It's also possible to activate a minimum level of points, by setting a *Minimum Time Factor* (between 0 and 1, as a floating point value). Setting the factor to 0.25, for example, tells the system that one quarter of the points will be gained by the user, if the question has been correctly answered but the time has run out. A factor of 0 tells the system to give no points when time's up, and 1 equals setting the time-based points calculation to false.

Single-Click

The above defined formula is sufficient for Single-Click questions.

Multiple-Click

As for Multiple-Click questions mostly two or more answer options are correct, and to avoid cheating, positive points are assigned only when all answer options are correctly checked (as they were authored to be correct).

Answer-Ordering

Each answer option can be assigned points (as for the other two types), so the user earns points for every correct position of an answer option. That means: if an option is authored to be for example on the 3^{rd} position, the user gets points if s/he places it correctly. For that specific item s/he may get 5 points, whereas for placing the 1^{st} item correctly s/he gains only 1 point. The reason for that is to be found in the authoring, because the author can freely specify the amount of points[1] for right or wrong answer-options.

[1]Points have to be specified as whole numbers, but can be positive or negative.

5.4.2 Simulation

In a similar way to some of the questions in the quiz component the user has to answer single-click questions:

1. Prioritise among cases: select ONE (earns points)
2. Choose a treatment option relating to the prioritised case: select ONE. Although here with only two possible answers there is a fifty-fifty chance for the user, the response also brings points according to the used time.

The basic formula here is:

$$P_{Scenario} = \sum_{Incident=1}^{4} (P_{Question} + P_{Incident} + P_{Hint})$$

Where $P_{Question}$ is taken from the quiz component's formula in subsection 5.4.1.

5.4.3 Rewards & Penalties

Users do not earn points for rewards or penalties. For more details on rewards and penalties see subsection 5.2.4 on page 42.

In advanced versions of the game, rewards could be extended by the possibility to reveal hidden material (CeLOs) to gain extra points. This idea is described in more detail in section 7.3 on page 71.

5.4.4 Hints

Hints may bring the user plus, minus or zero points. Again, the amount of points is up to the author.

5.4.5 Time-Aspect

Depending on the game's configuration, time is not only noticeable for giving answers, but also for gaining points. Setting the parameter `ANSWER_POINTS_TIME_BASED` to true, lets fast users gain more points than thoughtful ones, by taking the used time into account for the calculation of points.

5.4.6 Authoring

As mentioned before, the amount of points that can be gained by the user is not limited by the game template. However, the authoring interface could be implemented in the way that it is, for example, not possible to assign more than 2 points for a hint, while points for answer-options are limited to -3 to 8.

5.5 Non-Features

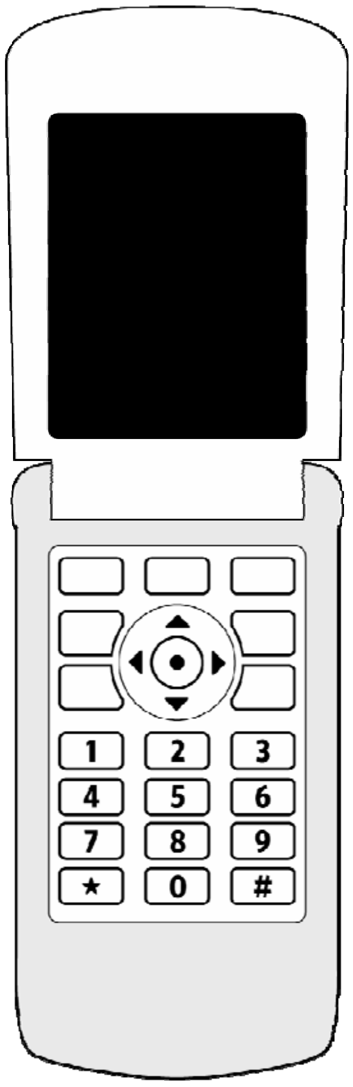
The aim of this section is to make clear what features will not be implemented, and why. Table 5.2 lists the most important of those features, someone might expect to be part of the application, and the reasons for not implementing them.

Table 5.2: Non-Features, that is, features someone may expect to be implemented, but won't for the described reasons.

Feature	Comment
Previous screen	There is no use case for letting the user access the previous screen/CeLO, and therefore it will not be implemented.
Support all available handsets	Although the defined minimum requirements are met by a wide range of mobile phones, we picked only a few of them for a prototypically implementation (see subsection 6.1.2), however, there still are a lot of differences between those devices.
Video support	Playback of video-snippets itself is not easy. Additionally it raises up a lot of other questions like "what (should) happen(s) if a device does not have (certain) video capabilities" or "videos reserve a fair amount of memory within a jar-file, that not all phones can cope with. Whereas streaming also is not an option because it still does not work well for mobile devices".

5.6 Controls

Figure 5.5 shows the keyboard layout for mobile phones without touch-screen. However, smartphones with a touch-screen, which are quite popular now, are also supported by the application. The player can use both: his/her stylus ,like s/he is used to, and the keyboard like it's defined in figure 5.5.



2 , ▲	Scroll up
8 , ▼	Scroll down
5 , ⊙	Select
6 , ▶	Next
4 , ◀	Joker selection
★	Menu

Figure 5.5: The controls layout on the user's device (independent of the game component)

Chapter 6

Template Implementation

6.1 Standards

6.1.1 API

As there is hardly any actual mobile phone that does not support it, and there come a lot of benefits with it, the chosen API is the *Java 2 Micro Edition*, based on MIDP2.0 and CLDC1.1.

6.1.2 Devices[1]

For commercial game developers there exist lists that mention about 400 different mobile devices that should be supported by an application. Because we lack the financial power of commercial game producers, we concentrate on just a few mobile phones:

- Sony-Ericsson M600i
- Sony-Ericsson P990i
- Sony-Ericsson v630i
- Nokia 6230i

Those devices are representatives for classes of mobile phones with different screen resolutions. Therefore, developing the application for those devices should enable us to support also other devices of those classes.

6.1.3 Screen Sizes

Game Template 1 *Ahead of the Game* will be optimized for the following selected screen sizes, that result from the above specified test-devices:

- 176x220
- 208x208
- 240x320

[1] A detailed description of the devices' features can be found here: http://www.gsmarena.com or here: http://www.j2mepolish.org/devices-overview.html

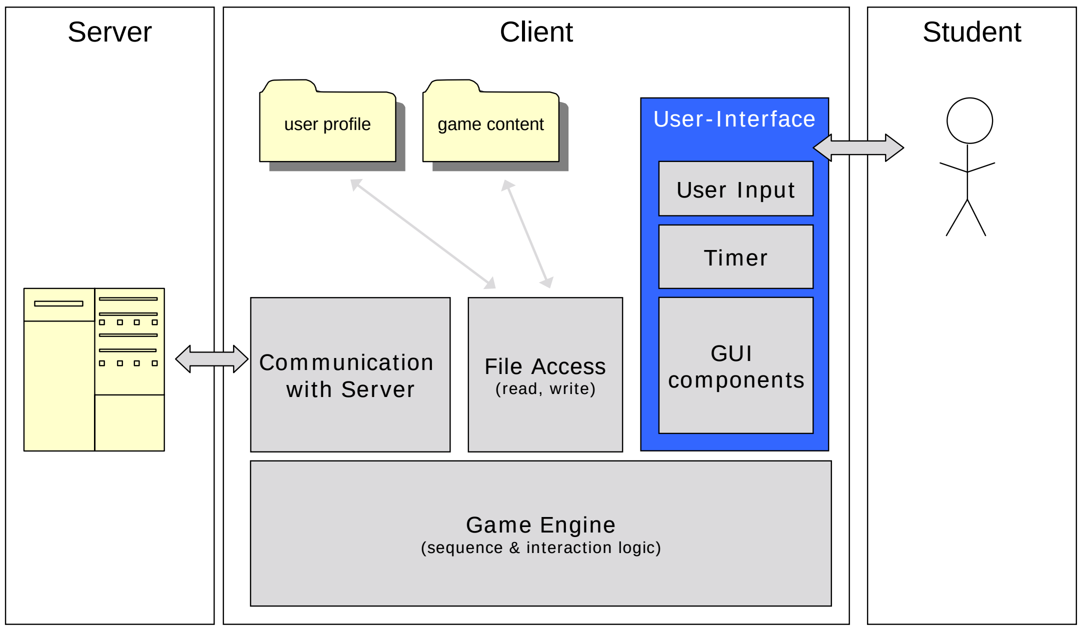

Server
Client
Student
user profile
game content
User-Interface
User Input
Timer
GUI components
Communication with Server
File Access (read, write)
Game Engine (sequence & interaction logic)

Figure 6.1: Overview of the whole system from a client's perspective

6.1.4 Multimedia Content

The playback of multimedia content like video and sound clips requires the Multimedia-API (MMAPI), which still is no guarantee that a certain format is supported. Therefore supporting multimedia content also asks for alternatives in cases where the playback of the content is not possible. Thinking of a preparation video that explains a current situation, a picture or a text is often not an option.

In view of all these problems we are not going to support multimedia content yet, but will try to keep the software design open for later implementations.

6.2 Architecture Overview

As portrayed in figure 6.3, that, in contrast to figure 4.1, focuses on template aspects, the author creates content using the web-platform (*Web-Site*). The so created data is stored in a central database (*DB*). The system automatically accesses the database, incorporates the content with the game template (*Game Creation Engine*) and makes the game available for download by students (*Deployment*). A separate component handles subsequent communication with the client handsets (*Game-Engine*).

Figure 6.2 shows the concept of how the game template is filled with content. The author creates a game concept that mainly consists of texts and graphics. This data is authored using, and stored in a database provided by, the web authoring tool. A pre-processor and compiling tool then incorporates the content provided by the author and the game template to an executable file.

In addition to figure 6.3 and figure 6.2, a combined view of them is presented in figure 6.1,

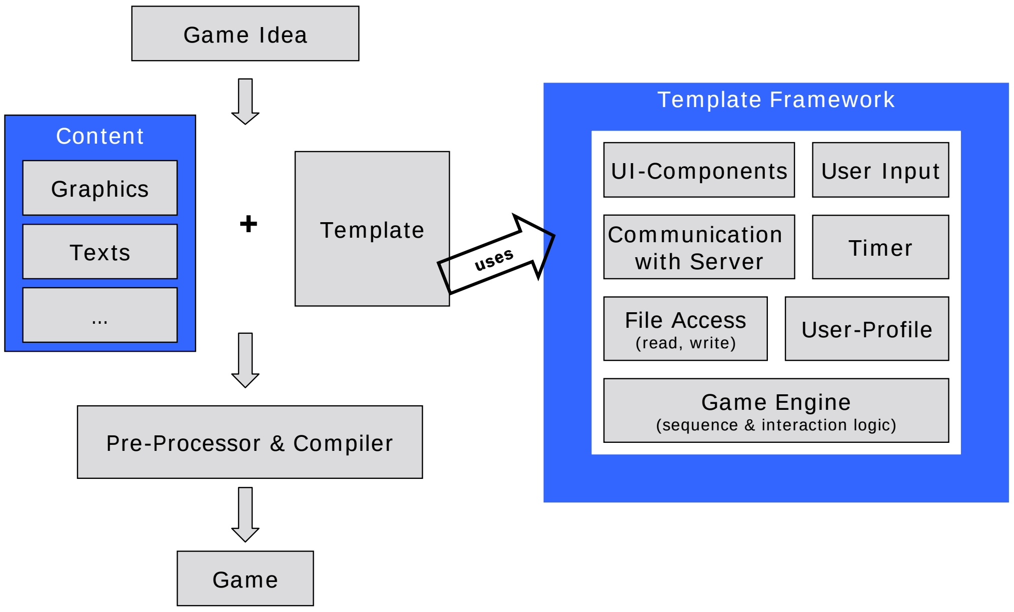


Figure 6.2: Schematic representation of the game creation process

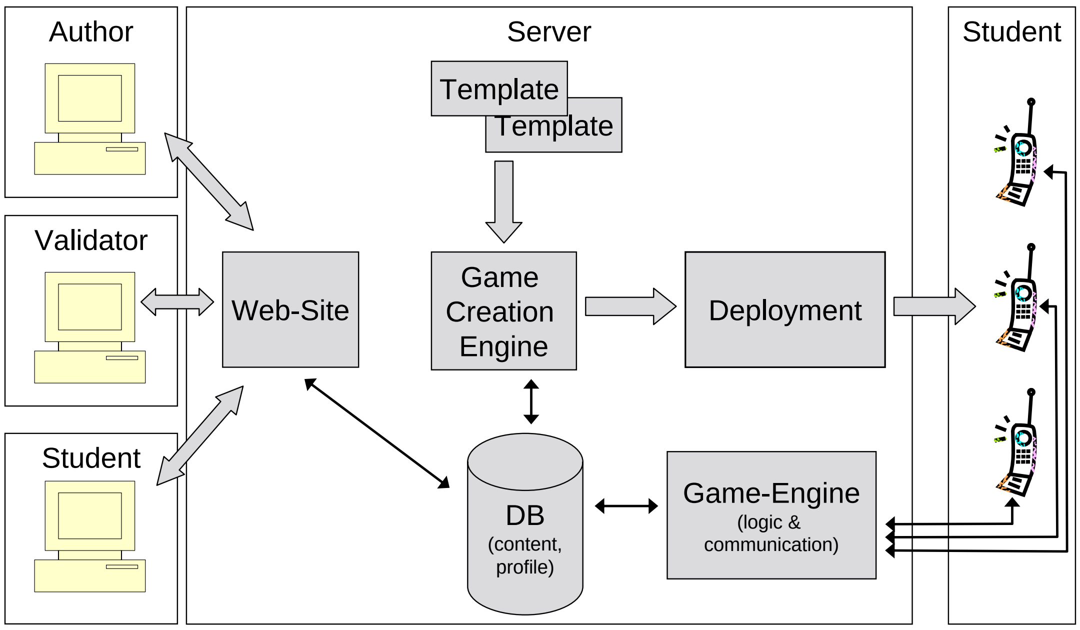


Figure 6.3: Overview of the whole system from a server's perspective

to highlight essential components of the game template in relation to the outside world.

6.3 Package Structure

The main package is `at.researchstudio.sat.mgbl`; subpackages are described in table 6.1.

Table 6.1: The application's package structure

Package	Description
`io`	Classes of the IO package take care of data-communication with the server. These connections are based on either SMS or HTTP.
`model`	The model classes mainly represent the CeLOs. In some cases, when it is necessary to store additional data like achieved points, instances of these classes get updated from within view or controller classes (e.g. `QuestionRepresentation`).
`model.userprofile`	The user's profile is, at game end, transferred to the server, where it's used to compute statistics including other games and game instances. The profile is also used in-game, to feed the action trigger, the self-assessment component, and others.
`store.dao.impl`	There are two main purposes of this package: reading and writing data to/from the database(s). A huge part is used for reading the game content from the provided data.xml file. Another, in contrast very small, part is used to write essential data to the mobile phone's RMS-Storage (to be used later for statistical purposes).
`store.dao.impl.gamedata`	All classes of this package are used to retrieve CeLO information from the used database (the data.xml file). The entry point for XML parsing is the class `AheadOfTheGameParser`, which initiates the XML parser, reads some general data, skips all data of previously solved, or future game levels, and finally passes the parser instance to other objects, which do the CeLO parsing.
`ui`	The package is dominated by subclasses of `AbstractRepresentation`, which support basic functionality for drawing content on the mobile phone's screen, and reacting on key or pointer events.

The application's package structure (continued)

Package	Description
`ui.framework`	The UI-Framework consists of elements needed to create a screen that is presented to the user. The most important elements are: • Scrollbar • Textbox • Checkbox-Group • Radiobutton-Group
`util`	This package provides, beside other classes, utils for mathematical, graphical and timing purposes. Additionally to those helpful functionality, the class `Properties` stores publicly accessible constants that are used to configure the application. These are for example constants describing the text-boxes' height and width, text snippets, boolean values to activate the debug mode or to shuffle answer options, and so forth.

6.4 Software Design

Figure 6.4 shows the most important packages and their elements of the client application. These elements are described in more details in the following sections.

6.4.1 Controller

The Controller is the central element of the application (on the left side in figure 6.4). It triggers the loading of new content, manages all loaded CeLO elements, and takes care of presenting each CeLO to the user.

Key-/pointer events

Keystrokes and pointer events are also handled by the Controller, if they are system-relevant (e.g. opening the menu); or they are forwarded to the current representation, which itself may forward it to one of its components.

Draw, Repaint

Whenever the MIDlet is called to draw the screen content, it forwards this call to the controller. It may draw items itself before, or after, further forwarding the request to the currently active representation.

When the state of an representation changed, e.g. because of user interaction, the representation may invoke the controller's `repaint()` method to trigger a fresh paint of the whole screen (which does simply a forward to the MIDlet's `repaint()` method).

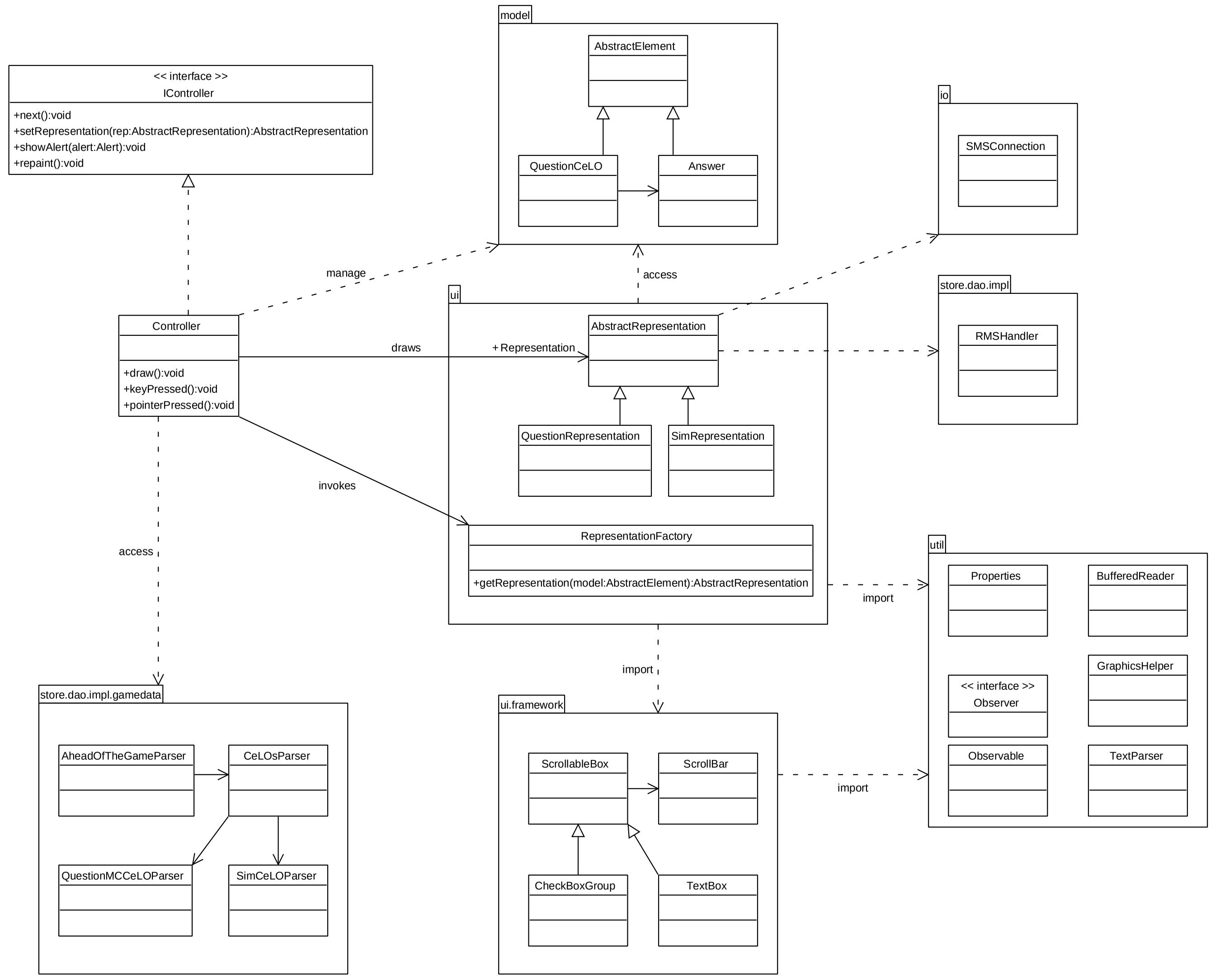


Figure 6.4: *Ahead of the Game* design overview (showing the most important representatives per package, only)

6.4.2 Threading

The usage of threads is limited to some occasions, e.g. the content parsing and simultaneously updating of the progress bar. In this example, also the observer pattern has been applied to manage updates.

While dealing with thread in the context of MIDlets, it is essential not to block the main thread (the one, the application is running in), because this would block the entire application and even key-strokes, repaints, or other events could not be processed — the whole application would hang.

6.4.3 XML-processing

Entry point for content parsing is always the `AheadOfTheGameParser` (in the left-bottom corner of figure 6.4). It instantiates the XML parser, reads some basic information (e.g. the joker definitions), and further forwards the parser instance to the `CeLOsParser`. The instance forwarding repeats several times, because the parser is divided into several pieces that are hierarchically organized.

If the `CeLOsParser` can't recognize the type of a CeLO element, the current XML element (and each sub-element) is ignored and an error message is printed on `stdout`.

Each parser requires a specific layout of XML elements, sub-elements and attributes, and may throw an exception if it is not met.

The used XML library is a minimal version of the freely available kXML 2 library[2]. The library is based on the Pull Parser API[3], which defines a new approach of reading XML documents, complementing *Document Object Model* (DOM) and *Simple API for XML* (SAX).

6.4.4 UI-Framework

Visual elements from the UI framework (see bottom of figure 6.4) are used to build up a representation. Standard J2ME libraries do not sufficiently support the required user interface design of the game, and even can't be extended to. Therefore the self made elements of this package build the basis of the whole application. The most important elements are:

- Scrollbar
- Textbox
- Checkbox-Group
- Radiobutton-Group

6.4.5 Representations

A representation combines several framework elements to one displayable screen (find some examples in the centre of figure 6.4). It manages the drawing of all components in the right order, as well as key and pointer events. Every representation is a subclass of `AbstractRepresentation`, which has basic implementations of the above described functionalities. Its main responsibilities can be classified in these categories:

[2]See: `http://kxml.sourceforge.net/`, accessed at 15.11.2007
[3]See: `http://xmlpull.org/`, accessed at 15.11.2007

- Background — e.g. a background image
- Foreground — e.g. a textbox that is shown in front of a background image
- Keystrokes- and touch-screen-events that are used to control the game. Each representation is responsible for handling it's own key- and touch-events.

6.4.6 Connection to the world

Data transfer to and from the game server comes in two flavours: SMS and HTTP. The user can choose his/her favourite protocol via the game settings (HTTP is faster, but in many countries still more expensive than SMS). The package including the communication classes is shown in the top-right corner of figure 6.4.

6.4.7 User Profiles

User profiles are stored in the device's RMS during the game. This is done for mainly two purposes: to access data (points, etc.) of previous mastered game levels (e.g. to calculate an overall statistic with the self-assessment element), and to transfer (overall) scoring information to the game server.

6.4.8 Utils

`MathHelper`, `GraphicsHelper`, classes to implement the observer pattern, and some others build a collection of useful tools. They are mostly used from within representation classes.

Properties

Configuration parameters of the whole application are centralized within this class (the Properties, as well as other utility classes are illustrated in the bottom-right corner in figure 6.4). For a later release version it's planned to outsource most of the properties into two distinct files: a configuration and a language file. The configuration includes e.g. the debug flag, parameters to configure the scoring system, and dimensions of bars, boxes, etc. The language file will contain all text snippets used by the game to create texts. It should easily be possible to switch from the standard language to another, by just replacing this one file.

6.5 Datamodel

figure 6.5 gives a good overview, of the different CeLO types and their interaction with each other. It also gives hints about how the application can be expanded, when needed.

The central element is the abstract class `CeLO` (shown in the centre of figure 6.5), where all other CeLO types are derived from it. Every CeLO may have one or more preparation CeLOs of any type, that are presented before the associated CeLO in sequential order. As every CeLO may have preparations, they may occur even on different levels (preparations of a preparation), although this is often not useful. However, the application does not count points that may result from preparation elements, irrespective of their level.

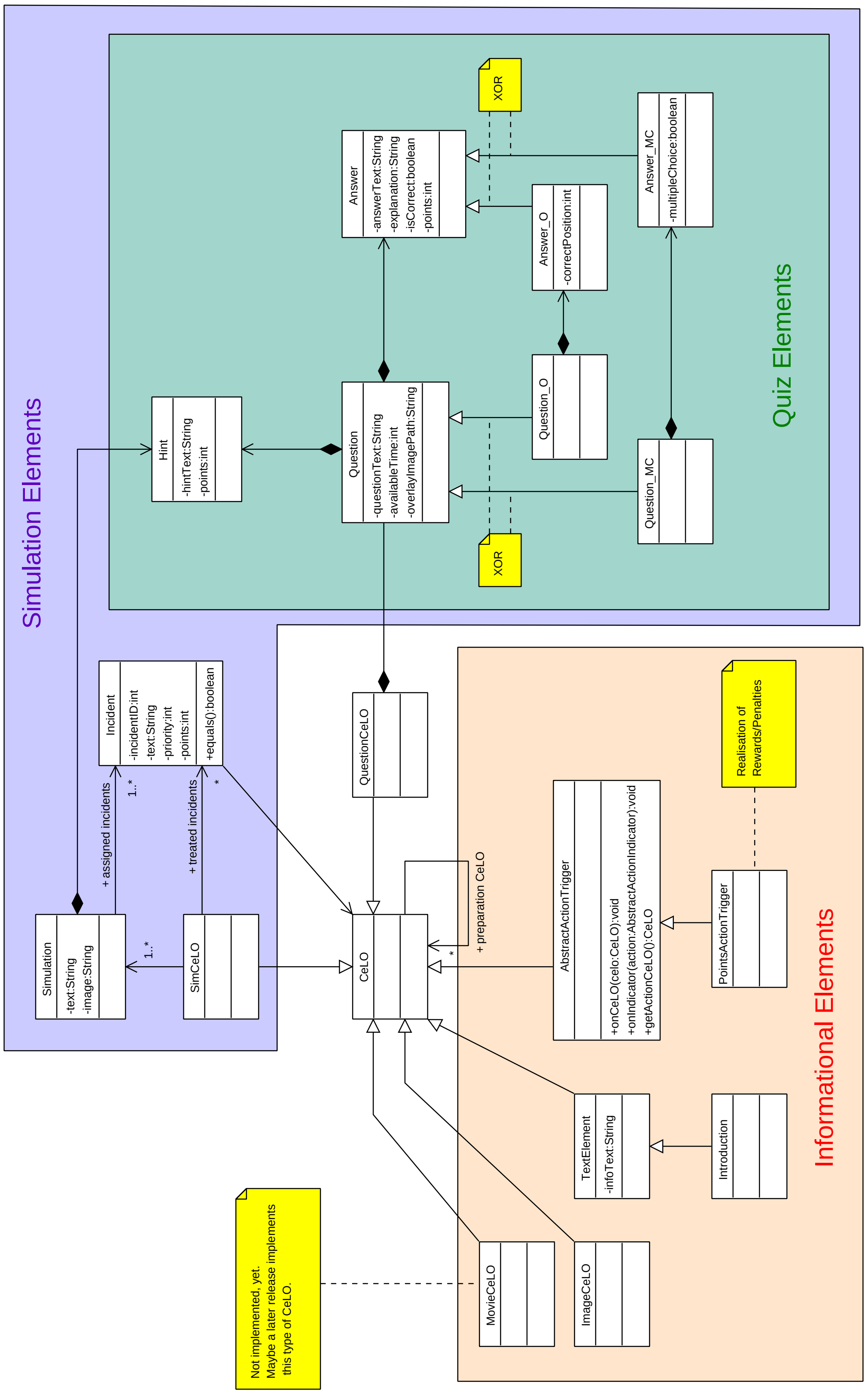


Figure 6.5: *Ahead of the Game* data model

Informational CeLOs are arranged in the bottom-left corner of figure 6.5. They have, in contrast to quiz or simulation elements, only informational character, what means that they do not interact with the user but rather present some kind of information.

On the right side of figure 6.5, all classes that are relevant for Question CeLOs are grouped together. To avoid co-variant problems, different subtypes of `Question` and `Answer` have been introduced.

Simulation elements are grouped in the upper area of figure 6.5, where the association from `Incident` to `CeLO` makes clear, that every CeLO can be used as incident-description. In contrast, the frame that claims the simulation area conform to the, in section 5.3 documented, specification of assigned questions, only. The difference of the specification and the more powerful implementation was the intention of those two visualisations.

6.5.1 Game Information

The following information can be authored for every game component:

- Time based point-calculation (yes|no)
- Permute Answers (show answers in a different order than authored)
- Number of available jokers for:
  - Hint
  - 50/50
  - Pause

6.5.2 General Information

A CeLO is the super-type for all learning objects and consists of the following data, which is inherited by all subtypes:

- Unique ID
- Corresponding Game-Level-ID

6.5.3 Quiz

A *Q-CeLO* is a subtype of a CeLO and adds the following data:

- Question
  - Unique Question ID
  - Text
- Answers, including for every answer:
  - Text
  - Correct = yes | no
  - Achievable points (can be negative)

- Short in-game explanation, why the model answer is correct[4].
- Hint + points (positive/negative) for using it
- Available time
- Overlay image

In accordance with the answer models employed, we define two subtypes of a Q-CeLO.

Q-CeLO-MK stands for two sub-classes of CeLOs, that are distinguished in the way, the user can select answer options. Therefore, this type extends a Q-CeLO with a Boolean value, which is used to distinguish between the two possibilities:

- Single-click (SK): the user picks one answer-option at a time — only one answer-option can be correct.
- Multiple-click (MK): one or more answer-options together form a correct response. Therefore, the author specifies one or more correct answer-options, the user has to match to gain all achievable points.

Q-CeLO-O the user has to bring the permuted answers in the correct ordering, what makes it essential to let the author define this order, and transfer this information to the client software. Therefore, an additional field is introduced, that stores the correct order for each answer-option.

6.5.4 Simulation (SimCeLO)

Each *SimCeLO* consists of typically four *Simulation*s, representing one "critical situation" as it develops over four points in time. For each simulation the author describes the situation at the current point in time, including a set of four *Incidents*, which also may have changed (as shown in figure 6.5):

- Simulation text
- Overlay image (optional)
- Hint (optional)
- A full set of Incidents, for each Incident must be authored:
  - Incident-ID
  - Unique-ID[5]
  - Priority
  - Points
  - A CeLO instance of any type (as shown in figure 6.5). Typically, a Question with the following characteristics is used for that purpose:

[4]It is possible to author additionally detailed end-game explanations, which is usually presented on the web-platform. Porting this feature to the client software could be part of future development.

[5]While the Unique-ID explicitly identifies one marked Incident, it's Incident-ID identifies a class of incidents, that represent the development of one incident over time.

  * Question text
  * Question image
  * 2 answer options
  * Hint
  * Time (in seconds)
  * etc.[6]

Typically, four incidents together form one situation, but the implementation allows n (where $n > 0$) incidents, which are different for n points in time. Please mind, that n deterioration levels requires a $n \times n$ matrix to be authored, which is a reasonable authoring effort, why we suggest that $n = 4$.

Example

Please find authoring examples in subsection 5.3.6 and appendix C.

6.5.5 Feedback

The user may answer a customisable question with free-text, which is sent to a special area on the server, where it's viewable by the author. The player may also cancel this dialogue in the case s/he has nothing to say, or does not want to pay for the data transfer. In this case, it's up to the teacher to interpret the missing notion.

6.5.6 Self-Assessment

The self assessment can be configured by changing the following parameters:

- Self-Assessment Text (e.g. "How do you think you have done?")
- n times (usually $n = 4$):
  - Threshold (e.g. 0.7, what means 70%)
  - Threshold-text (e.g. "good" or "excellent")

Several threshold and according text values create a classification, the user must use to categorize him/herself. The system is implemented very flexible, to make it adaptable to locale standards (for example US American use a different grading system than German schools).

6.5.7 Information (InfoCeLO)

This kind of representation consists of a background image and user specified text (e.g. for introduction purposes). The several subtypes differ in the instance of background image they use and where the text is positioned on the screen. Subtypes are:

- Info (background: flipchart)
- Introduction (background: boss)
- Reward / Penalty (background: various depending on reward/penalty)

[6]For further details see subsection 6.5.3.

6.5.8 System generated Information/Representations

There exist a few representations that cannot be customized by the author, but are generated by the application. These are:

- Gamelogo
- Startup-Screen
- Loading-Screen
- Highscores-Screen

6.6 Settings, Configuration

Settings configure the application. This can be done at several levels, but always has to be supported by the system and therefore specified before the implementation takes place. Most important settings are those configurable by users (authors and players), because they require an intuitive (graphical) user interface.

The settings listed below are divided into categories presenting roles of the people who are responsible for changing them (according to the above mentioned levels).

6.6.1 Player

The player can change the following game settings via the game's user interface:

- Sounds (on, off)
- Server-connection (SMS vs. HTTP, incl. time-outs and retries)
- Server-address (could be provided during deployment automatically, but can be changed by the user)
- user-name and password

6.6.2 Author

Via the web-platform, the author can change the following data:

- Jokers
- Meta-information (title, topic, sub-topic, etc.)
- Content + Timing + Scoring
- Security (allowed users/user-groups)

6.6.3 Developer

Developers make a pre-configuration of the game, by changing the following settings during the implementation phase:

- Activate debug mode
- System generated text-snippets
- File-names (e.g. config-file)
- Image-paths
- Minimum and maximum supported screen resolutions and Java APIs (an error message will be generated if the resources don't match the requirements)
- Colours, sizes, icons, images, ...

6.7 Server Communication

As described in subsection 5.1.7, it was planned to add some competition with other users to the game. To reduce costs, a communication flow between server and client, visualised in figure 6.6, has been developed. Because the mentioned competition is not implemented yet, the diagram could be reduced to the communication-flow at game end. However, for future work, the whole planned communication flow is described in the next sections, where it should be obvious which of them are not relevant for the current implementation.

6.7.1 Distribution

First of all the user needs to download the game to his/her mobile phone (described on page 23), which is the first interaction with the *mGBL*-server.

6.7.2 Game Start

At game start, the application sends a HELLO message to the server to trigger the actual highscores to be sent.

The server responds to the HELLO message with the current highscores for all non-finished levels as well as the avatar id (i.e. the bosses sex) of the users in that list.

6.7.3 Game end

At game end, the current user profile[7] will be transferred to the server, what requires the user's authorization.

The actual ranking will be calculated on the server and transferred to the client, so it only needs to present it on the screen (without doing any calculations).

[7]For detailed information concerning the user profile, please see section 6.10.

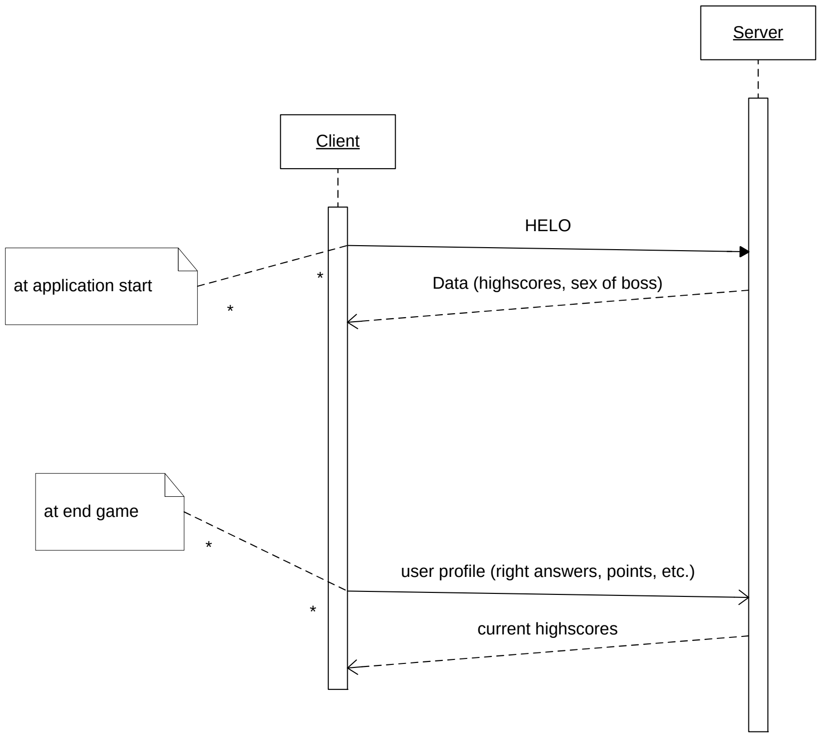

Server
Client
HELO
at application start
Data (highscores, sex of boss)
*
*
at end game
*
user profile (right answers, points, etc.)
*
current highscores

Figure 6.6: Schematic client-server communication

6.7.4 Feedback[8]

As for all other server communications (e.g. transfer of free-text answers, see subsection 6.5.5), the user specifies (via the settings dialogue) whether to use SMS or HTTP based connections. The user composed feedback is sent immediately to the server.

6.7.5 Component-Synchronization

When playing a hybrid game composed of the quiz and the simulation component, the simulation must not be played before the quiz has been successfully completed. To ensure the predefined order, the server must ensure that the second component is not downloadable before the first has been finished.

6.8 File Structure

The game-data file stores content concerning all CeLO types of the game. Although differently specified[9], it's possible to mix Simulation-CeLOs and other types of CeLOs within one game as well one level. However, this should not be considered as a drawback, but as a feature — the application allows content structures like specified, and even more complex ones.

The most important characteristics of the content file are:

- Levels are numbered consequently - there is no missing level.
- Level 0 indicates a general introduction.
- There must be at least one answer option for each question (independently of the question's type)[10].
- Optional parameters may be omitted.

The exact definition of the content file, specified in XML's *Document Type Definition* (DTD), can be found in the appendix (listing A.1).

6.9 Highscores

At game end, the ranking will be presented to the user. The presentation is intelligent in the way, that the user sees e.g. not only the top 5, but always his/her own rank, beside some key-ranks. Let's, say for example, the user becomes 17th, s/he will see the top three, his/her own rank, and the last place.

It is planned to implement the possibility to keep the user updated even after game end, so that s/he can subsequently ask for his/her ranking via a game's menu entry. In either case, users have access to their scores via the web-platform.

[8]For further information, please see subsections 5.2.3 and 6.5.5

[9]See chapter 5

[10]There should be two or more answer options, otherwise multiple-choice makes no sense.

6.10 User Profiles

The information stored in the profile and sent to the server is:

- User-ID (indirectly by MSISDN, or via user-name & password credentials)
- Overall achieved points

Advanced versions of the game may transfer more detailed statistical information, such as on a per-CeLO basis:

- CeLO-ID
- Selected answer-options (for Q-CeLOs)
- Correctness of answers
- Used jokers
- Used time

6.11 Game Deployment

The deployment tool will be seamlessly integrated into the server platform, and is responsible for creating the downloadable jar-files out of the authored data and the game template for different target devices. Schematic representations of the game creation process, in different views, are shown in figures 4.1, 6.1 and 6.2.

Chapter 7
Evaluation

7.1 Hype or Tripe

Georg Ströhlein tries to identify characteristics of mobile learning scenarios that identify them whether as hype or tripe [Str07]. He admits that mobile learning has the potential of being a hype, when it is ready for everyday's usage by relying on widely spread technology, or a tripe, when it can not. The consent of his work was, that it heavily depends on the realization of the learning process or model, whether it is the one or the other. Based on continuous user trials, we tried to identify and eliminate weaknesses of *Ahead of the Game*. Overall we can now say that our game template is on the hype-side, although there are some aspects that need to be improved in future versions. Therefore, user's concerns are documented in the next chapter, while conclusions of these outcomes as well as ideas for improvements that don't come from user trials are summed up in section 7.3.

7.2 Feedback from User Trials

User trials have been conducted at different formal levels and in different states of development to address different questions.

First of all some kind of slide-show with static mock-ups has been implemented for mobile phones. These mock-ups were pictures presenting the basic elements of the user interface to get information about dimensions, colours, and arrangement of elements on a device's display.

An early implementation of the game was then developed, mainly to work on PC emulators for the 1^{st} *user trials*. It has been presented to many potential users of both targeted user groups – students and teachers – in several countries to get an impression of how they would accept the game idea as well as the user interface. About 150 participants (13 teacher, the rest students) attended the 1^{st} user trials online, where the majority liked the presented concept of *Ahead of the Game*.

After an elaborate enhancement of the learning model[1] and its implementation, a small, unofficial user trial was conducted to check the clarity and comprehensibility of the new game concept. This survey has been conducted at the *Microlearning Conference*, which was considered appropriate and meaningful to test a mobile game. The very useful hints for improvements could be found on three levels:

[1]The quiz component was enhanced and set into a common context with the newly established simulation component.

- User interface and its design elements: the font size was considered too small, while the separation of question and answer-options into two distinct boxes was confusing the participants.
- Storyline and appropriateness of system responses: participants were irritated because of some similar questions and simulation-situations of the game[2]. We learned from these responses that the preparation of the content and its sequence is at least as important as the user interface or the learning theories that are the basis of the game.
- Interaction of software and hardware: too long loading times, and, in contrast, too fast responses on selecting or deselecting answer options via the touch screen have been reported, as well as too long responses when clicking the *next* button. Unfortunately, this is the level we have the least impact on, because it mainly concerns the underlying hardware, which is sometimes quicker or slower, depending on the used device.

2nd *user trials* have, again, been conducted within the context of the *mGBL* project. Content, in the field of e-commerce, has been newly created and translated into the language of the hosting country (Slovenia). Participants where 6 teachers and 5 students, where all but one have been provided with mobile phones. One teacher brought her own cell phone, where the game has been installed onto. Results of the these user trials showed that the user interface could be further improved (or reconsidered). Because there were hardly negative comments regarding the content or its storyline and the hardware-software interaction (see above), we conclude that these issues have been resolved. Instead, we received a lot of feature requests, where the most prominent were the support of video-clips and a multi-player mode.

The user interface It has been reported that not enough content is shown at a time and the scoring was not successfully explained (especially that of multiple-answer questions), where the latter point was mainly raised by students. A discrepancy between teachers and students was found on comments concerning the controls of the game. While students were satisfied with the controls, although they needed some time to get familiar with the given device, teachers found that controls were not clear enough. Although, teachers did not pay much attention on the game design, they, as well as students, recommended to add animations as some kind of variation.

Content and storyline were considered properly, although some students mentioned they would not pay any attention if texts were longer, but select an answer randomly.

The simulation was flavoured by teachers, who thought it is a good way to make people familiar with real life situations in a simulated, thus safe, environment. They also had the idea to support some kind of decision trees, where the decision of one situation puts the player in another one that s/he has to cope with again[3].

[2]Indeed, the content was not prepared carefully enough, so that it seemed to some participants to have a déjà vu.

[3]For example, the user is put in the situation to help one of four people that are hurt in a car accident, s/he picks one and finds him/herself in a situation where to cope with four injuries (e.g.: head wound, broken leg, swollen eye, stomach-ache) of that particular person.

Other issues concerned mainly multimedia content (video clips) and a multi-player version of the game. Students also demanded some kind of statistical analysis tool to help teachers identify possible weaknesses of particular students, student groups or the game content.

The male boss was considered inappropriate by most participants in several user trials. Therefore we added the possibility to switch, at the beginning of the game, to a female boss character. Although this feature was implemented in an early stage of development, people still reported that they didn't like the picture. It seems that it is really essential to replace this character with a more appropriate one.

7.3 Ideas for Improvements

The game conception phase and the user trials produced some ideas for improvements and nice-to-haves. To sketch the potential of the implemented prototype, these are listed below, starting with important improvements, followed by more exceptional ideas.

A better explanation of points is essential before making the game available for the broad public. User trials showed (see section 7.2), that it is not clear enough to users, why they get a certain amount of points. Because additional plus or minus points may be gained by using a joker, these circumstances must also be explained sufficiently.

The scoring model must be improved to make it more acceptable for students. User trials showed that they are not comfortable with getting 0 points for a partially right multiple-answer question. Therefore, the scoring model of this CeLO type needs to be reconfigured in a way that partially right answers bring points, too. On the one hand, this may seduce some students to cheat — checking all answer options, a random selection, or a certain pattern (e.g., options 1, 3 and 4) may lead to reduced, but still positive scores. On the other hand, a reworked scoring model combined with accurate authoring may reduce cheating to a minimum.

Animations were considered as both fun and distraction, within our 2nd user trials. Because of this divergent meanings, implemented animations have to be both discreet and appealing at the same time. However, animations are, as all of us know, an integral part of all games, because they entertain and attract, and are indispensable, therefore.

Statistical analysis of the participants' progress and its availability on the server platform for the student and his/her teacher. Tracking each step the player makes and transferring this statistical data to the server is a good possibility for teachers to determine weaknesses of individual players, or the whole group. Instead of finding weaknesses in the group of students, such a tool could be used to make weaknesses in the authored content or the whole curricular visible.

Reveal hidden features or bonus material by extraordinary good or bad scores, interesting (but not necessarily correct) answers, or by a system generated random event adds some value of uncertainty and engages players to keep playing and reach (again) good scores [SZ03]. Examples are medals for outstanding scores, that are visible to all players via the web-platform, or additional CeLOs that may bring bonus points or free hints for questions to come.

A Multi-player mode can be implemented in a direct or indirect way. The indirect way makes the game round based, where one game level correlates to one round. At the beginning of each round, the client software requests the current highscores from the server and presents the leading person as the player's boss who is challenging him/her. Additionally the complete highscore list (including the player's score) may be shown at the end of each round. This kind of multi-player mode was initially planned, but not implemented, because of time constraints due to changing requirements (see subsection 5.1.7). A direct multi-player mode is much more complicated, without negating the advantages of time and space independence, that people love so much about mobile devices. One possible approach is to make one player, who has finished the game already, a validator of one or more others. S/he receives the player's challenge and his/her solution and has to judge the solution as right or wrong. A correct judgement brings the validator bonus points, while the player earns additional points for a correct solution with an incorrect judgement. However, the group of participating people must be large enough to ensure a proper validation service, and thus a continuous game flow.

Chapter 8

Conclusion

In this work I have presented basics, and especially benefits of mobile learning (section 2.3), game based learning (section 2.2), and the combination of both areas (section 2.4). While the main benefits of game based learning is to be found in the learner's motivation and engagement with the topic, mobile learning enables students to follow their own preferences in learning while keeping focused for a longer period.

Based on state of the art development technology (*JavaME*), this work focuses on the specification and implementation of a prototypical mobile learning game for mobile phones. The whole architecture is object and performance oriented, to address both, maintainable source code, and the user's comfort of fast system responses, respectively.

User tests showed that we (participants of the EU project *Mobile Game-Based Learning*) are on the right track with our implementation, but it is a long road to go, until an attractive, engaging and long lasting mobile learning game will result from our efforts. Particularly, a game *template* is even more laborious to develop, because it has not only to be attractive and engaging, but must be applicable to a broad spectrum of different, often orthogonal, contents. The list of requirements does not end at the client application, but requires an equally high qualitative authoring tool to get the content together with the template. The tool has to be balanced between an intuitive user interface and the power of a programming language, tight restrictions to enforce high quality content and the possibility to follow own ideas.

Concluding, I believe, technology will become even better and in a few years from now mobile phones will have the power of a today's game console, not only talking of its processing power but also of the provided development tools. Therefore, we have to continue working now, to find ways to motivate learners of any age, to have mature learning models ready, when technology is.

Chapter 9

Bibliography

[Abt87] Clark C. Abt. *Serious Games.* University Press of America, Mai 1987.

[ASS71] Elliot M. Avedon and Brian Sutton-Smith. *The Study of Games.* John Wiley, New York, 1971.

[Att04] Jill Attewell. Mobile technologies and learning. Technical report, Technology Enhanced Learning Research Centre, 2004.

[Com] Java Community. Java specification requests. `http://www.jcp.org/en/jsr/all`, accessed at 10.10.2007.

[DRL96] John V. Dempsey, Karen Rasmussen, and Barbara Lucassen. The instructional gaming literature: Implications and 99 sources. Technical Report 96-1, University of South Alabama, 1996.

[Ess06] Sebastian Esser. *Inside Mobiles – Software für Mobiltelefone.* Vdm Verlag Dr. Müller, 2006.

[Hla06] Peter Hlavac. Innovative user interfaces for accessing music on mobile devices. Master's thesis, TU Wien, 2006.

[HS07] Richard Harrison and Mark Shackman. *Symbian OS C++ for Mobile Phones*, volume 3. Wiley & Sons, June 2007.

[JPa] J2ME-Polish. J2ME-Polish name clarification. `http://www.j2mepolish.org/faq.html#name`, accessed at 10.10.2007.

[JPb] J2ME-Polish. Nokia 6230i memory limitations. `http://www.j2mepolish.org/devices/Nokia/6230i.html#memory`, accessed at 10.10.2007.

[LJD07] Richard Leggett, Scott Janousek, and Weyert DeBoer. *Foundation Flash Applications for Mobile Devices.* Computer Bookshops, January 2007.

[Mal81a] Thomas W. Malone. Toward a theory of intrinsically motivating instruction. *Cognitive Science*, 5(4):333–369, 1981.

[Mal81b] Thomas W. Malone. What makes computer games fun? *Byte*, 6:258–277, December 1981.

[Mat] School Matters. Mobile phones mobile minds. Available at: `http://video.google.com/videoplay?docid=1043898959196049305`, accessed at 1.11.2007.

[PKD04] Maja Pivec, Anni Koubek, and Claudio Dondi. *Guidelines for Game-Based Learning.* Pabst Science Publishers, 2004.

[Pre01] Marc Prensky. *Digital Game-Based Learning.* McGraw-Hill Education, 2001.

[Rod04] Leon Rodin. E-learning to m-learning: an investigation into the potential for content conversion. In Jill Attewell and Carol Savill-Smith, editors, *Mobile learning anytime everywhere*, pages 171–176, 2004. Available at: `http://www.m-learning.org/archive/docs/MLEARN%202004%20book%20of%20conference%20papers.pdf`, accessed at 10.10.2007.

[Sch07] Klaus-Dieter Schmatz. *Java Micro Edition.* Dpunkt Verlag, 2. edition, 2007.

[SHB04] Liina Stotz, Gabriela Hoppe, and Prof. Dr. Michael H. Breitner. Interaktives Mobile(M)-Learning auf kleinen Endgeräten wie PDAs und Smartphones. IWI Discussion Paper Series 12, Institut für Wirtschaftsinformatik, Universität Hannover, August 2004. Available at: `http://ideas.repec.org/p/ifw/iwidps/iwidps12.html`, accessed at 10.10.2007.

[Str07] Georg Ströhlein. Mobile learning using mobiles: Hype or tripe? In *Neue Trends im E-Learning*, pages 1–15, Juni 2007.

[Sun01] Todd Sundsted. J2ME grows up – New specs signal a growth spurt for Java 2 Platform's smallest edition, May 2001. `http://www.ibm.com/developerworks/java/library/j-j2me/`, accessed at 15.11.2007.

[SZ03] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals.* The MIT Press, November 2003.

[TB04] John Traxler and Nell Bridges. Mobile learing – the ethical and legal challenges. In Jill Attewell and Carol Savill-Smith, editors, *Mobile learning anytime everywhere*, pages 203–207, 2004. Available at: `http://www.m-learning.org/archive/docs/MLEARN%202004%20book%20of%20conference%20papers.pdf`, accessed at 10.10.2007.

[Wika] Wikipedia. Adobe Flash Lite. `http://en.wikipedia.org/wiki/Flash_Lite`, accessed at 15.11.2007.

[Wikb] Wikipedia. JAR file format. `http://en.wikipedia.org/wiki/JAR_(file_format)`, accessed at 10.10.2007.

[Wikc] Wikipedia. Java community process. `http://en.wikipedia.org/wiki/Java_Community_Process`, accessed at 15.11.2007.

[Wikd] Wikipedia. Java version history. `http://en.wikipedia.org/wiki/Java_version_history`, accessed at 10.10.2007.

[Wike] Wikipedia. Java virtual machine: Bytecode verifier. `http://en.wikipedia.org/wiki/Java_Virtual_Machine#Bytecode_verifier`, accessed at 10.10.2007.

[Wikf] Wikipedia. MIDlet. `http://en.wikipedia.org/wiki/Midlet`, accessed at 10.10.2007.

[Wikg] Wikipedia. Mobile learning. http://en.wikipedia.org/wiki/Mobile_learning, accessed at 1.11.2007.

[Wikh] Wikipedia. .Net compact framework. http://en.wikipedia.org/wiki/.NET_Compact_Framework, accessed at 15.11.2007.

[Wiki] Wikipedia. Symbian os. http://en.wikipedia.org/wiki/Symbian_os, accessed at 15.11.2007.

[Wikj] Wikipedia. Unreal game engine. http://de.wikipedia.org/wiki/Unreal_Engine, accessed at 01.11.2007.

[Wikk] Wiktionary. Game. http://en.wiktionary.org/wiki/game, accessed at 01.11.2007.

[Wor] WordNet. Wordnet 2.1 search-term 'game'. Available at: http://wordnet.princeton.edu/obtain, accessed at 01.11.2007.

[YD04] Paul Yao and David Durant. *.Net Compact Framework Programming with C (C Sharp)*. Addison-Wesley Longman, May 2004.

[Zaca] Erik Zachte. Wikipedia access statistics. http://stats.wikimedia.org/EN/TablesUsagePageRequest.htm, accessed at 10.10.2007.

[Zacb] Erik Zachte. Wikipedia usage statistics. http://stats.wikimedia.org/EN/ChartsWikipediaEN.htm, accessed at 10.10.2007.

[ZHKH07] Weishan Zhang, Dong Han, Thomas Kunz, and Klaus Marius Hansen. Mobile game development: Object-orientation or not. In *Computer Software and Applications Conference, 2007. COMPSAC 2007 - Vol. 1. 31st Annual International*, pages 601–608, July 2007.

Appendix A

File Format

The format of the content file, specified in DTD-Syntax, can be found in listing A.1.

Listing A.1: The format of the content file, specified in DTD-Syntax

```
<!ELEMENT AheadOfTheGame (Jokers?, Level+)>
    <!ELEMENT Jokers (Joker+)>
        <!ELEMENT Joker EMPTY>
            <!ATTLIST Joker
                type CDATA #REQUIRED
                number CDATA #REQUIRED
            >

    <!ELEMENT Preperation (QuestionMCCeLO|QuestionOCeLO|SimCeLO|Info|
        Introduction|Image|ImageText|QuestionFreeText|SelfAssessment|
        Reward)+>

    <!ELEMENT Level (QuestionMCCeLO|QuestionOCeLO|SimCeLO|Info|
        Introduction|Image|ImageText|QuestionFreeText|SelfAssessment|
        Reward)+>
        <!ATTLIST Level
            level CDATA #REQUIRED
        >

        <!ELEMENT Reward (Preperation?, History, Action+)>
            <!ATTLIST Reward
                id              CDATA           #REQUIRED
            >

            <!ELEMENT History EMPTY>
                <!ATTLIST History
                    history         (all)       'all'
                >
<!--                        history         (all|level|last)     'all' -->

            <!ELEMENT Action (QuestionMCCeLO|QuestionOCeLO|SimCeLO|Info
                |Introduction|Image|ImageText|QuestionFreeText|
                SelfAssessment)>
                <!ATTLIST Action
                    comp            (lt|gt|eq)          'eq'
                    value           CDATA               #REQUIRED
                >
        <!ELEMENT QuestionFreeText (Preperation?)>
            <!ATTLIST QuestionFreeText
                id              CDATA           #REQUIRED
                question        CDATA           #REQUIRED
                answer          CDATA           #IMPLIED
            >

        <!ELEMENT SelfAssessment (Preperation?, SelfAssessmentThreshold
            +)>
            <!ATTLIST SelfAssessment
                id              CDATA           #REQUIRED
                question        CDATA           #REQUIRED
            >

        <!ELEMENT SelfAssessmentThreshold EMPTY>
            <!ATTLIST SelfAssessmentThreshold
                threshold       CDATA           #REQUIRED
                text            CDATA           #REQUIRED
            >

        <!ELEMENT Info (Preperation?)>
            <!ATTLIST Info
                id              CDATA           #REQUIRED
                text            CDATA           #REQUIRED
            >

        <!ELEMENT Image (Preperation?)>
            <!ATTLIST Image
                id              CDATA           #REQUIRED
                picture         CDATA           #REQUIRED
            >

        <!ELEMENT ImageText (Preperation?)>
            <!ATTLIST ImageText
                id              CDATA           #REQUIRED
                picture         CDATA           #REQUIRED
                text            CDATA           #REQUIRED
                overlay         (true|false)    'true'
            >

        <!ELEMENT Introduction (Preperation?)>
            <!ATTLIST Introduction
                id              CDATA           #REQUIRED
                text            CDATA           #REQUIRED
            >

        <!ELEMENT QuestionMCCeLO (Preperation?, AnswerMC+, Hint?)>
            <!ATTLIST QuestionMCCeLO
                id              CDATA           #REQUIRED
                question        CDATA           #REQUIRED
                multipleChoice  (true|false)    'false'
                picture         CDATA           #IMPLIED
            timeAvailable   CDATA           #REQUIRED
        >

        <!ELEMENT AnswerMC EMPTY>
            <!ATTLIST AnswerMC
                answer      CDATA           #REQUIRED
                explanation CDATA           #REQUIRED
                points      CDATA           #REQUIRED
                correct     (true|false)    'false'
            >

        <!ELEMENT Hint EMPTY>
            <!ATTLIST Hint
                hint    CDATA   #REQUIRED
                points  CDATA   #REQUIRED
            >

    <!ELEMENT QuestionOCeLO (Preperation?, AnswerO+, Hint?)>
        <!ATTLIST QuestionOCeLO
            id              CDATA           #REQUIRED
            question        CDATA           #REQUIRED
            picture         CDATA           #IMPLIED
            timeAvailable   CDATA           #REQUIRED
        >

        <!ELEMENT AnswerO EMPTY>
            <!ATTLIST AnswerO
                answer      CDATA           #REQUIRED
                explanation CDATA           #REQUIRED
                points      CDATA           #REQUIRED
                order       CDATA           #REQUIRED
            >

    <!ELEMENT SimCeLO (Preperation?, Simulation+)>
        <!ATTLIST SimCeLO
            id  CDATA #REQUIRED
        >

        <!ELEMENT Simulation (Incident+, Hint?)>
            <!ATTLIST Simulation
                id              CDATA           #REQUIRED
                question        CDATA           #REQUIRED
                picture         CDATA           #IMPLIED
            >

        <!ELEMENT Incident (QuestionMCCeLO|QuestionOCeLO|SimCeLO|
            Info|Introduction|Image|ImageText|QuestionFreeText|
            SelfAssessment|Reward)>
            <!ATTLIST Incident
                incident-id     CDATA           #REQUIRED
                id              CDATA           #REQUIRED
                text            CDATA           #REQUIRED
```

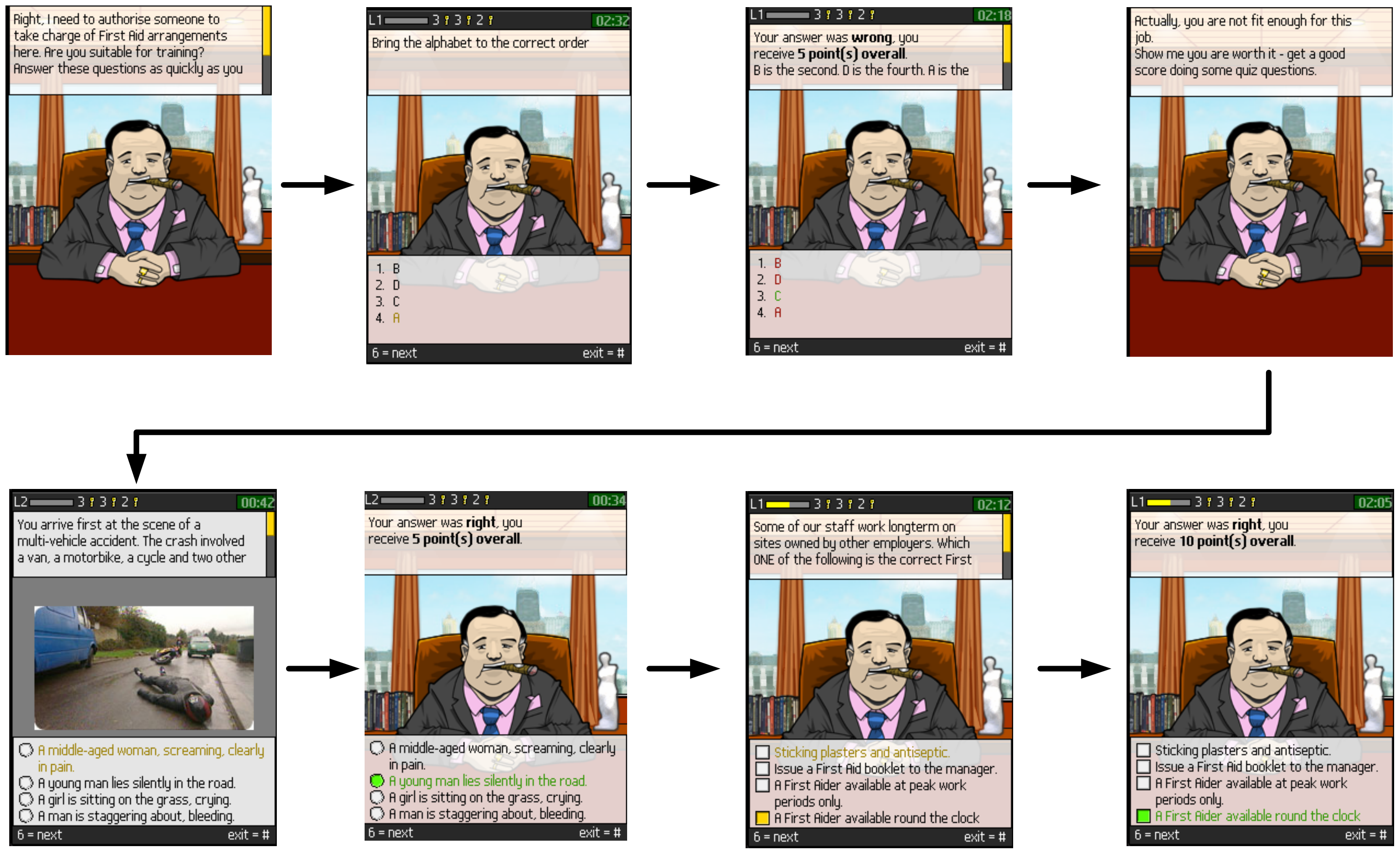
```
          prio            CDATA           #REQUIRED
          points          CDATA           #REQUIRED
        >
```

Appendix B

Game Examples

Figures B.1 and B.2 exemplary show the game-flow of a quiz and a simulation component, respectively. The first (figure B.1) presents the three different question types (Ordering, Single-Click and Multiple-Click) and the system-reaction on wrong and right user input.

A look at figure B.2, gives an impression of how a simulation, and especially deterioration, works. For clearness, only two of four possible incidents are presented here, while the descriptions of the different situations have been omitted for the same reason. At the two remaining incidents, you can see how they are deteriorated over time, where it might occur – like for the second incident ("unconscious man") – that a patient dies. From a complete 4×4 matrix of all incidents only one element per row is presented to the user, depending on the deterioration level.


Right, I need to authorise someone to take charge of First Aid arrangements here. Are you suitable for training? Answer these questions as quickly as you

L1 3 ! 3 ! 2 ! 02:32
Bring the alphabet to the correct order
1. B
2. D
3. C
4. A
6 = next exit = #

L1 3 ! 3 ! 2 ! 02:18
Your answer was **wrong**, you receive **5 point(s) overall**.
B is the second. D is the fourth. A is the
1. B
2. D
3. C
4. A
6 = next exit = #

Actually, you are not fit enough for this job.
Show me you are worth it - get a good score doing some quiz questions.

L2 3 ! 3 ! 2 ! 00:42
You arrive first at the scene of a multi-vehicle accident. The crash involved a van, a motorbike, a cycle and two other
○ A middle-aged woman, screaming, clearly in pain.
○ A young man lies silently in the road.
○ A girl is sitting on the grass, crying.
○ A man is staggering about, bleeding.
6 = next exit = #

L2 3 ! 3 ! 2 ! 00:34
Your answer was **right**, you receive **5 point(s) overall**.
○ A middle-aged woman, screaming, clearly in pain.
● A young man lies silently in the road.
○ A girl is sitting on the grass, crying.
○ A man is staggering about, bleeding.
6 = next exit = #

L1 3 ! 3 ! 2 ! 02:12
Some of our staff work longterm on sites owned by other employers. Which ONE of the following is the correct First
☐ Sticking plasters and antiseptic.
☐ Issue a First Aid booklet to the manager.
☐ A First Aider available at peak work periods only.
☐ A First Aider available round the clock
6 = next exit = #

L1 3 ! 3 ! 2 ! 02:05
Your answer was **right**, you receive **10 point(s) overall**.
☐ Sticking plasters and antiseptic.
☐ Issue a First Aid booklet to the manager.
☐ A First Aider available at peak work periods only.
☑ A First Aider available round the clock
6 = next exit = #

Figure B.1: Exemplary game-flow of a quiz component.

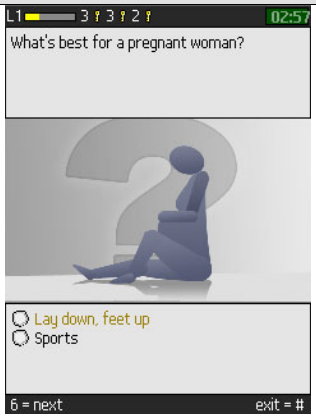
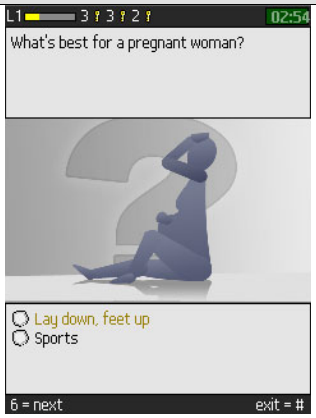
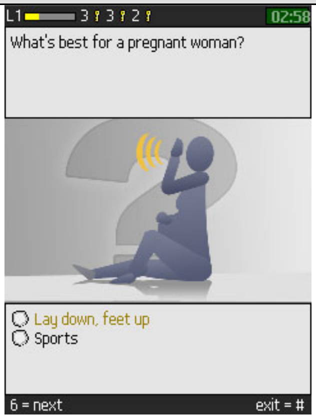
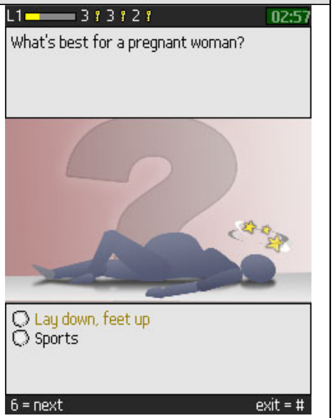
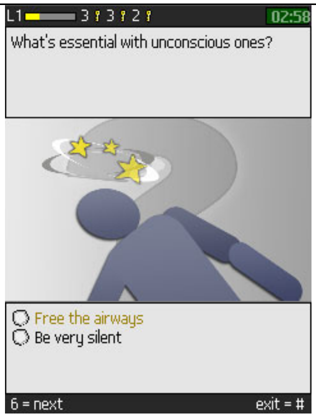
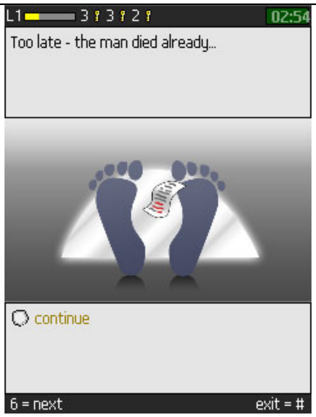
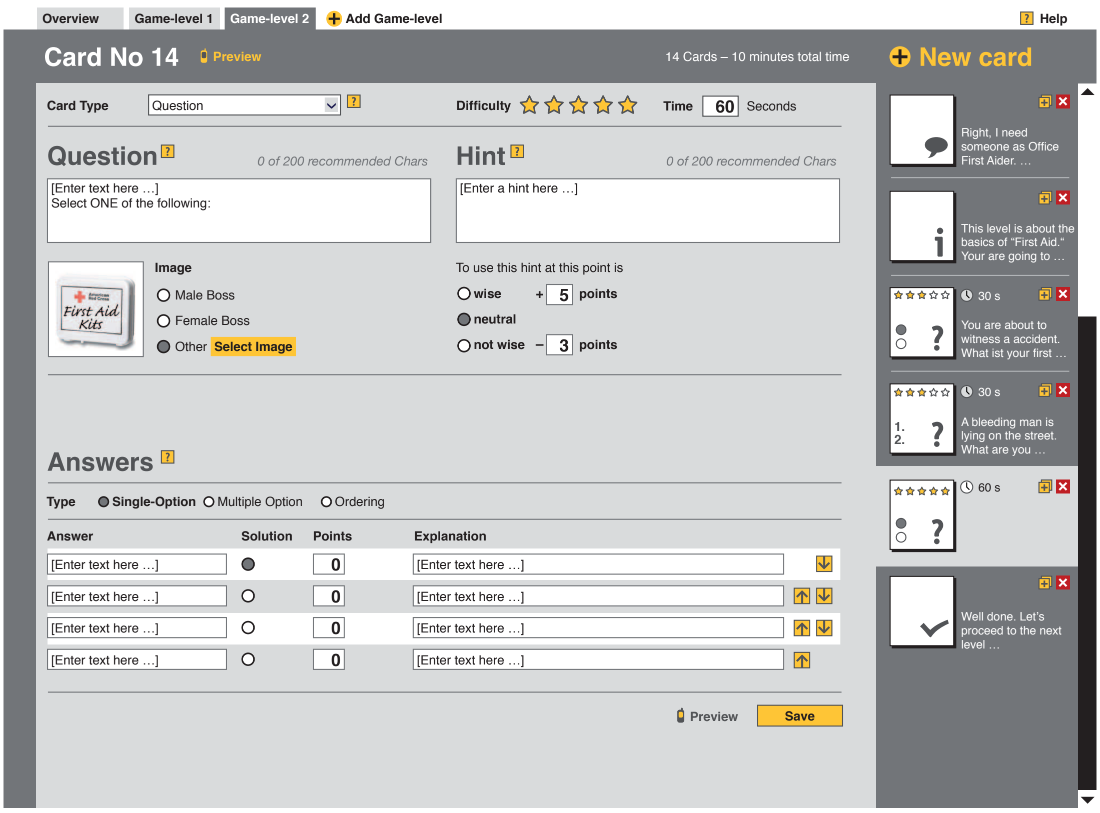
	Start	Deterioration 1	Deterioration 2	Deterioration 3
Question 2 “pregnant woman”	 L1 3 3 2 02:57 What's best for a pregnant woman? Lay down, feet up Sports 6 = next exit = #	 L1 3 3 2 02:54 What's best for a pregnant woman? Lay down, feet up Sports 6 = next exit = #	 L1 3 3 2 02:58 What's best for a pregnant woman? Lay down, feet up Sports 6 = next exit = #	 L1 3 3 2 02:57 What's best for a pregnant woman? Lay down, feet up Sports 6 = next exit = #
Question 4 “unconscious man”	 L1 3 3 2 02:58 What's essential with unconscious ones? Free the airways Be very silent 6 = next exit = #		 L1 3 3 2 02:54 Too late - the man died already... continue 6 = next exit = #	

Figure B.2: Two incidents of a simulation component, and their deterioration over time.

Appendix C
Authoring Interface

To give an impression of how the authoring may work, I want to present some exemplary design mock-ups here. Figures C.1 and C.2 show the authoring interface, mainly used for the quiz component. The difference between both can be seen at the bottom of the graphics, where the answer-options are edited — the one uses radio-buttons, the other check-boxes. On the right side the author is provided with an overview, of the so far edited elements, while the edited game-levels are shown at the top. Usually, the authoring starts by choosing the card-type (SC, MC, Simulation, etc.), and editing the question text. Beside a custom image (left side of figure C.1), the author may specify a hint including bonus or penalty points for using it. The difficulty-level at the top and the radio-buttons below the hint, help unexperienced users at their first steps.

Figure C.3 presents the authoring design for the simulation component. As you can see, the navigational elements are the same as for the quiz component, while the content creation is a bit more complex. Therefore, it is possible to fold elements to a small, representative overview, to be able to keep an overview of the whole picture. The several deterioration levels are arranged from left to right (separated by dotted lines), while details of each level are arranged vertically. In the example (figure C.3), the first incident at the first deterioration level has been already edited. To make the user interface look not too overcrowded, an incident can have a Single-Click Question with exactly two answer-options, only.



Overview | Game-level 1 | Game-level 2 | Add Game-level | Help

Card No 14 Preview

14 Cards – 10 minutes total time

New card

Card Type: Question

Difficulty ☆☆☆☆☆ Time 60 Seconds

Question

0 of 200 recommended Chars

[Enter text here …]
Select ONE of the following:

Hint

0 of 200 recommended Chars

[Enter a hint here …]

Image

Male Boss

Female Boss

[✓ Other] Select Image

To use this hint at this point is

wise + 5 points

[✓ neutral]

not wise – 3 points

Answers

Type: [✓ Single-Option] Multiple Option Ordering

Answer	Solution	Points	Explanation
[Enter text here …]	●	0	[Enter text here …]
[Enter text here …]	○	0	[Enter text here …]
[Enter text here …]	○	0	[Enter text here …]
[Enter text here …]	○	0	[Enter text here …]

Preview | Save

Right, I need someone as Office First Aider. …

This level is about the basics of "First Aid." Your are going to …

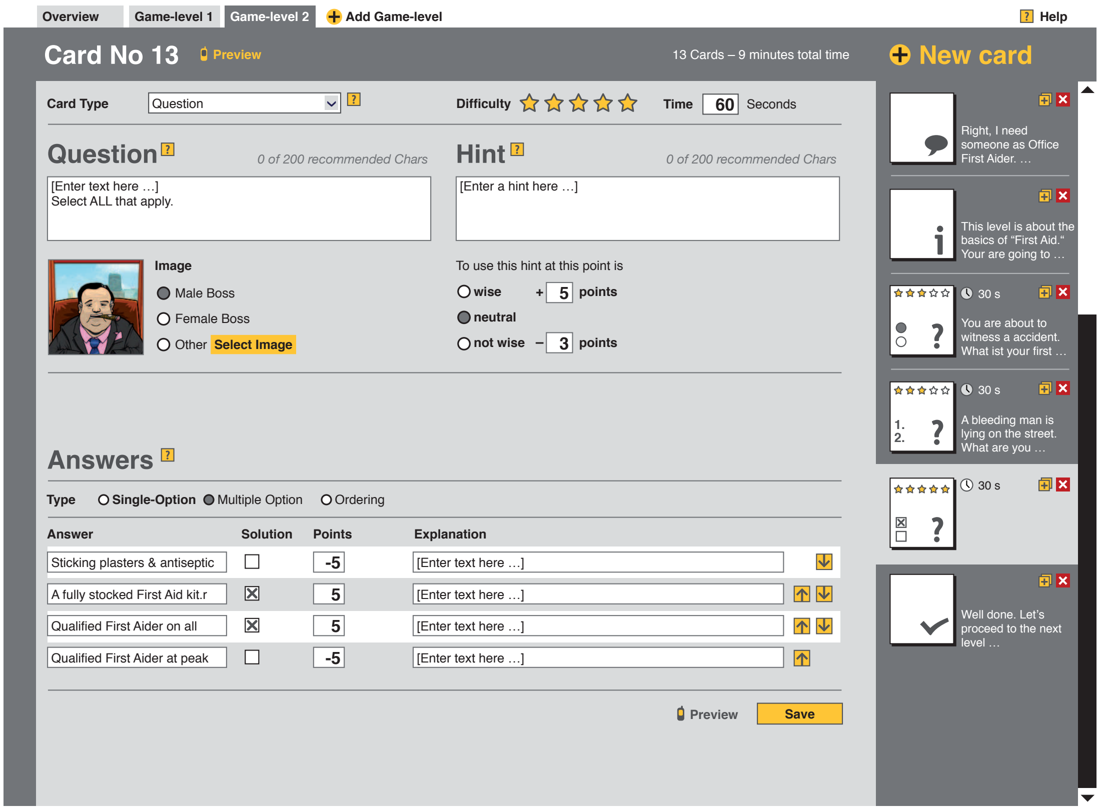
30 s — You are about to witness a accident. What ist your first …

30 s — A bleeding man is lying on the street. What are you …

60 s

Well done. Let's proceed to the next level …

Figure C.1: Authoring card for a Single-Click Question.



Card No 13 Preview

13 Cards – 9 minutes total time

New card

Card Type: Question

Difficulty ☆☆☆☆☆ Time 60 Seconds

Question

0 of 200 recommended Chars

[Enter text here ...]
Select ALL that apply.

Hint

0 of 200 recommended Chars

[Enter a hint here ...]

Image
[✓ Male Boss] Female Boss Other Select Image

To use this hint at this point is
wise + 5 points
[✓ neutral]
not wise – 3 points

Answers

Type: Single-Option [✓ Multiple Option] Ordering

Answer	Solution	Points	Explanation
Sticking plasters & antiseptic	☐	-5	[Enter text here ...]
A fully stocked First Aid kit.r	☒	5	[Enter text here ...]
Qualified First Aider on all	☒	5	[Enter text here ...]
Qualified First Aider at peak	☐	-5	[Enter text here ...]

Preview Save

Right, I need someone as Office First Aider. ...

This level is about the basics of "First Aid." Your are going to ...

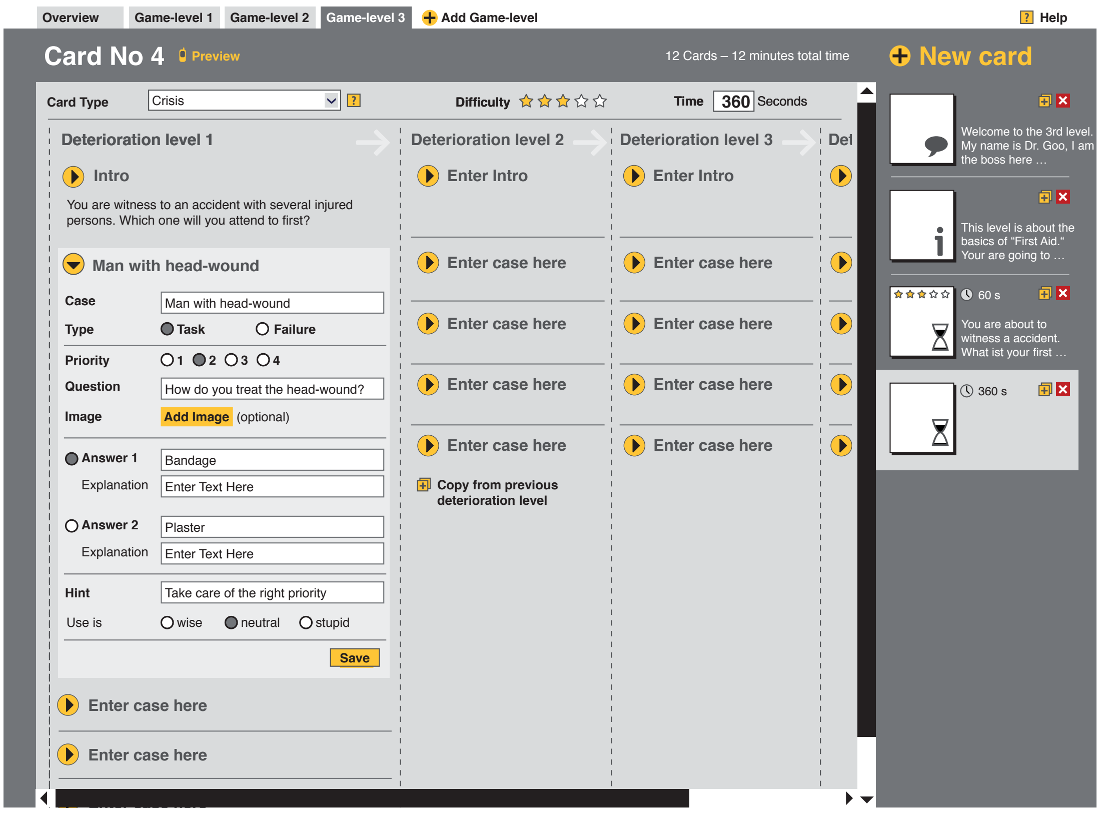
30 s
You are about to witness a accident. What ist your first ...

30 s
1.
2.
A bleeding man is lying on the street. What are you ...

30 s

Well done. Let's proceed to the next level ...

Figure C.2: Authoring card for a Multiple-Click Question.



Overview | Game-level 1 | Game-level 2 | Game-level 3 | Add Game-level | Help

Card No 4 Preview

12 Cards – 12 minutes total time

New card

Card Type: Crisis
Difficulty ☆☆☆☆☆
Time: 360 Seconds

Deterioration level 1

Intro

You are witness to an accident with several injured persons. Which one will you attend to first?

Man with head-wound

Case	Man with head-wound
Type	[✓ Task] Failure
Priority	1 [✓ 2] 3 4
Question	How do you treat the head-wound?
Image	Add Image (optional)
[✓ Answer 1]	Bandage
Explanation	Enter Text Here
Answer 2	Plaster
Explanation	Enter Text Here
Hint	Take care of the right priority
Use is	wise [✓ neutral] stupid

Save

Enter case here

Enter case here

Deterioration level 2

Enter Intro

Enter case here

Enter case here

Enter case here

Enter case here

Copy from previous deterioration level

Deterioration level 3

Enter Intro

Enter case here

Enter case here

Enter case here

Enter case here

Det

Welcome to the 3rd level. My name is Dr. Goo, I am the boss here …

This level is about the basics of "First Aid." Your are going to …

60 s

You are about to witness a accident. What ist your first …

360 s

Figure C.3: Authoring interface for a simulation (partially collapsed fields to provide a panoramic view).

Appendix D

Test-Devices

The used test devices and their key characteristics are presented in figures D.1 to D.4.

Sony Ericsson
Which of these items DOES NOT belong in a First Aid kit?
2 sterile eye pads
A course of antibiotics
1 pair of disposable gloves

Sony-Ericsson m600i

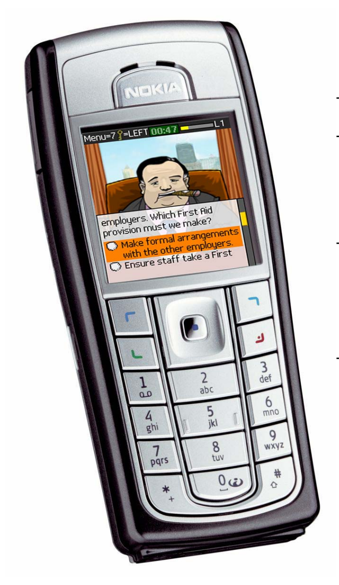
OS	Symbian OS v9.1, UIQ 3.0
Interactivity	TFT touchscreen QWERTY keyboard 240 x 320 pixels
Connectivity	GPRS Bluetooth 2.0 Infrared Port
Java	CLDC 1.1 MIDP 2.0 WMAPI 2.0 MMAPI

Figure D.1: Specification of used test device

Sony-Ericsson v630i

OS	Proprietary
Interactivity	176 x 220 pixels 256k colours
Connectivity	GPRS Bluetooth 2.0
Java	CLDC 1.1 MIDP 2.0 WMAPI 2.0 MMAPI

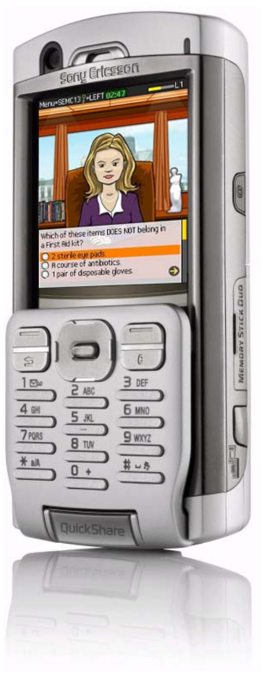
Figure D.2: Specification of used test device



Nokia 6230i

OS	Proprietary
Interactivity	240 x 320 pixels 65k colours
Connectivity	GPRS Bluetooth 1.2 Infrared Port
Java	CLDC 1.1 MIDP 2.0 WMAPI MMAPI 1.1

Figure D.3: Specification of used test device



Sony-Ericsson P990i

OS	Symbian OS v9.1, UIQ 3.0
Interactivity	TFT touchscreen QWERTY keyboard 240 x 320 pixels
Connectivity	GPRS Bluetooth 2.0 Infrared Port
Java	CLDC 1.1 MIDP 2.0 WMAPI 2.0 MMAPI

Figure D.4: Specification of used test device