VU UNIVERSITY AMSTERDAM

FACULTY OF SCIENCES

M A S T E R T H E S I S

to obtain the title of

Master of Science

of the VU University Amsterdam
COMPUTER SCIENCE - MULTIMEDIA

by

PETER PEERDEMAN

Intelligent Tutoring in Educational Games

Thesis Advisor: prof. dr. A. ELIËNS

Internship Supervisor: B. WEIJ, MA

Second Reader: prof. dr. M.C.A. KLEIN

February - July 2010

Acknowledgments

Contents

CHAPTER 1

Introduction

Serious games are getting more serious every day. However, in the Netherlands there aren't a lot of serious games available that offer innovative ways of educating young children. This is where the young company "Mijn naam is Haas" is trying to make a difference. By creating a game that really involves the user in an enjoyable virtual story that offers spoken dialog on several topics such as nature and food, the game tries to expand the users vocabulary knowledge.

By acquiring a government funding from the Maatschappelijke Sectoren & ICT, Mijn naam is Haas has been enabled to upscale their products to a broader market of schools and to further develop their products in order to better suit the educational market. One of the most important enhancements of the products is the implementation of an Intelligent Tutor System (ITS). This system ensures that the game dynamically adjusts its difficulty to the current player, offering content to a wide range of children with different vocabulary skills.

During my internship at the Mijn naam is Haas company I have helped to realize the ITS through different stages of the ITS project. From a basic vision document and several literature studies on Intelligent Tutor Systems that were already available, I have helped to design the conceptual models, to create the software and database architecture and implemented the server side back end of the ITS. During this process, I have gathered much experience in software engineering and working together with many people with diverse kinds of expertise to create even more advanced serious games.

In this thesis I will discuss several topics about Intelligent Tutoring Systems in general that have a direct relation to the work I have done on the Intelligent Tutor System of the game "Mijn naam is Haas" (MniH). While answering these contextual and technical questions, the research methods and gathered information will be discussed as well. The second part of the thesis consists of a detailed description of the development process of the ITS along with constructed diagrams that clarify the design choices made while engineering and developing the ITS.

In the following sections I will first give a brief introduction to the contextual and technical questions that will be discussed in this thesis. The contextual questions correspond to chapter 3 containing the background research based on existing material about Intelligent Tutor Systems.

Chapters 4 to 7 contain the technical documentation of the engineering and implementation process. The following part of this introductory chapter offers a thesis structure overview in which all the chapters of the thesis are described. The last part is a short reading guide that points the different target audience groups to the right sections of this thesis.

1.1 Contextual questions

The most important question that will be answered throughout chapter 3 is *"What is an intelligent tutor system?"*. To correctly answer this question, the intelligent tutor system concept is explained through several abstract models and relevant categorization techniques.

The following sections of chapter 3 address the question *"How does a game dynamically adjust its difficulty?"*, explaining certain difficulty progression and dynamic difficulty adjustment mechanisms used in games to automatically adjust the difficulty of the game to the skills of the player.

The last contextual question answered in chapter 3 is *"How can we create a vocabulary difficulty scale?"*, giving some insight into the terminology of vocabulary expansion teaching techniques and word list difficulty assessment used throughout the rest of the thesis.

1.2 Technical questions

The most specific issues addressed in this thesis are *"How can we create conceptual models of vocabulary expansion techniques"* and *"How can we convert these conceptual models to computational models?"*. The first issue is addressed in chapter 4, detailing the method of acquiring domain knowledge and constructing a vision on the system. The second issue is described in chapter 5, describing the computational models that comply to the conceptual models we created.

The less technical questions *"What features for the new version of MniH are needed to comply to the business plan?"* and *"How can an intelligent tutor system be engineered and implemented using known, proven methods?"* are introductory questions that really helped me out in the start of the internship to fully understand the subject and to get a grip on the different parts of work that had to be done. These questions are answered in chapter 4, by summarizing the goals and methods used in this project.

After these realizations were made and forged into my own wording of the ITS for MniH, we had to think about the distribution of work across the team of people that is working on this game. This resulted in the question *"How can the development of an intelligent tutor system be distributed in a cross disciplined creative team"?* that really determined the scope of the project and to helped me to acknowledge the value of all the individual team members.

The question that concerned most different kinds of expertise within the team was *"How can an intelligent tutor system be implemented to improve the educational effect of educational games?"*. I will elaborate on this in in chapter 5. This question resulted in a lot of discussions with the Expertise Centrum Nederland, which is closely involved in the feedback loop of the project.

The final part of the internship concerned the implementation of the designed intelligent tutor system into the overall game structure which led to the question *"How can an intelligent tutor system be integrated in an existing video games code structure"*. This issue is addressed in chapters 6 and 7 describing both the architecture and implementation of the system.

1.3 Thesis structure

This section gives a short description of each thesis chapter to give an overview of the available documentation. Chapter 2 gives a brief, illustrated introduction of the serious game Mijn naam is Haas to give a clear context for the intelligent tutor system. Using some annotated screen captures the gameplay and educational content of the game are explained.

Chapter 3 contains a literature research of intelligent tutor systems, an explanation of what tutor systems are, an example of dynamic difficulty adjustment, some background information on vocabulary expansion and information about the word lists to support the discussions on educational material later on in this thesis. Chapter 4 contains the vision and goal of the ITS in the MniH game, along with the methods used to elicit the requirements and conceptual models.

Chapter 5 contains the rough conceptual models of the ITS components, including the difficulty scaling, vocabulary tutoring, scoring and progression tracking. The architecture used to strictly define these conceptual models into implementable algorithms are described in chapter 6. The actual implementation of the backend side of the MniH ITS is depicted in chapter 7 using several software flow diagrams.

Chapter 8 gives a personal and professional evaluation of the approach, discussing the generalizability, pitfalls and future trends that could further improve the work that was done. Finally, chapter 9 gives a conclusion of the thesis, answering the posed contextual and technical questions and summarizing the results of the internship project.

The detailed scenario's, diagrams and other details are included in the appendices.

1.4 Reading guide

This thesis is written with a broad target audience in mind. The following groups are uniquely identified with their possible interests in this thesis and pointed to the right chapters:

Educational game developers To learn about the possibilities of implementing an intelligent tutor system. Chapters 2, 4, 5, 8.

Non-educational game developers To examine possibilities to launch their game on the educational market. Chapters 4, 8.

Mijn naam is Haas general employees To get a general idea of the ITS in MniH. Chapters 4, 5.

Mijn naam is Haas developers To read information on the development of the ITS. Chapters 6, 7, C.1.

Researchers Find out about the multitude of educational possibilities of intelligent tutoring in interactive media. Chapters 3, 8, 9.

Supervisors To examine the scientific value of this thesis and to provide feedback. Chapters 1, 2, 3, 8, 9.

All other readers interested in a brief overview of the contents and results of this thesis are advised to read the summary and conclusion in chapter 9.

CHAPTER 2

Mijn naam is Haas game overview

Mijn naam is Haas is a serious game in which the ingenuity of children is challenged by letting them draw objects into the world of main character Haas. By providing a rich, contextualizing spoken dialog in the native language of the student, the game unites educational vocabulary expansion with an immersive story environment.

This chapter gives a brief illustrated overview of the current Mijn naam is Haas gameplay mechanics and the educational goal of the game to create some context for the development of the ITS for this game.

2.1 Gameplay

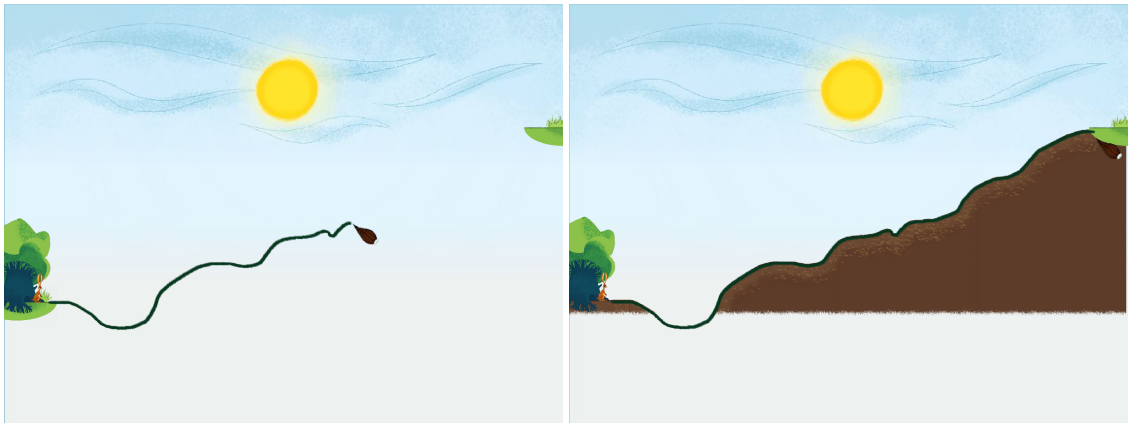

Figure 2.1: Left: Drawing the landscape line in Mijn naam is Haas. Right: resulting landscape.

The game starts with the main character "Haas", a hare who wants to go somewhere, depending on what title of the series is being played. In this example, Haas wants to find the greenest place in the game world. Before the game starts, Haas asks the student to draw the landscape. By drawing a line from the left to the right of the screen (figure 2.1), the student creates the game world, on which the upcoming conflicts will take place.

Figure 2.2: Main character Haas meets his friend Sofie.

After the landscape is drawn, Haas meets Sofie (figure 2.2), an intelligent female duck that helps Haas understand conflicts and helps him to contextualize words. Acting as a friend,

Sofie is able to offer extra audible information without needing any on-screen text or visual help. To avoid confusion, all game characters only address each other and only the Haas addresses the student directly, often with an incentive to do something.

Figure 2.3: Drawing a large line from the ground in the world of Mijn naam is Haas results in a tree being placed at the place where the line was drawn.

"Besides drawing the landscape for Haas to walk on, the user can draw objects, like trees and clouds, within the landscape. The drawings of the student affect the emerging story in realtime." [9]. For example, the student can draw a large line from the ground into the sky to produce a tree that is placed on that location (figure 2.3). When an object is drawn, one of the characters in the world will respond with spoken dialog about that object, to give the player more incentive to be creative and draw more.

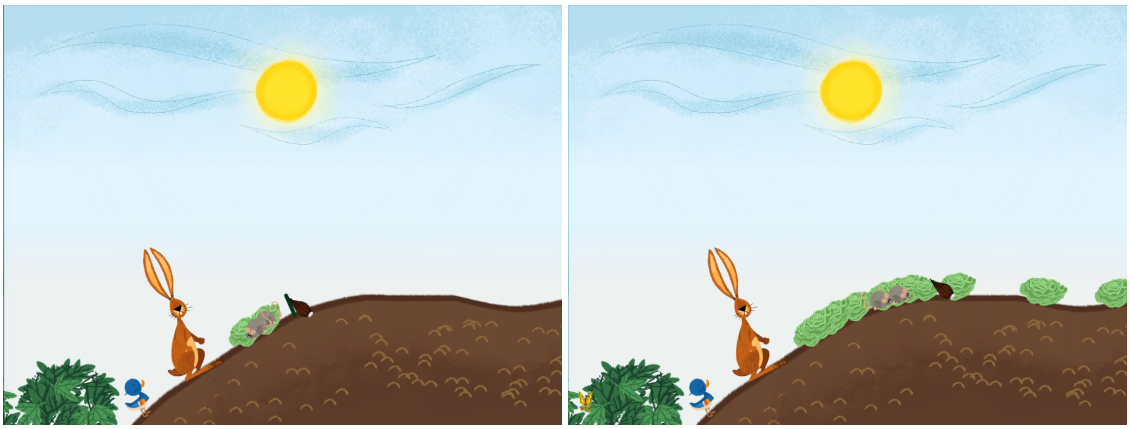

Figure 2.4: Because the voles want crops, the user has to draw small lines from the ground to draw crops into the world to solve the problem.

A gaming session, which takes about fifteen minutes, is split up into several conflicts, alternated by periods of free drawing. Conflicts are always chosen by the game engine based on certain game environment parameters to create a suitable situation. A conflict involves another character, who has a problem and needs certain objects to be drawn by the student to solve the problem. For instance, the voles are hungry and want crops on their field (figure 2.4). To solve this problem, the student has to draw small lines from the ground up to the sky in order to produce crops, after which the final dialog is played and Haas continues his journey to the next conflict.

2.2 Educational goal and content

The educational goal of this game is to achieve vocabulary expansion by immersing the student into a creative world. The method of teaching is quite like reading out a picture book, but because the student is interactively involved in the activity and has full attention to the subject matter, students better remember the content. In addition, the students fine motor skills are developed by manipulating the mouse or pen when a drawing tablet peripheral is used. Furthermore, the general creativity and imagination of students are stimulated.

The current educational content consists mostly of spoken dialog on different subjects with accompanying visualizations (e.g. visual representations of objects). Several game titles have been produced that all have a separate theme, such as a winter game, spring game and a title that takes place in autumn, so as to match the themes used in elementary schools. Each theme has different words associated with them that are explained and shown in the game through objects and dialog.

However, until now there haven't been any real educational guidelines for writing content and the writers just use their imagination and common sense to produce educationally justified content. With the introduction of the ITS, guidelines were developed for using themes, target words and sufficiently contextualizing dialogs. More information on these subjects can be found in chapter 5.

Furthermore, all available content is currently dispensed to the whole target group, including students from age 3 to 7. On this point, the ITS creates a big opportunity to specifically offer content to individual students tailored to their skills. This forces the storywriters to write content in a specific, broader range of difficulties and enables them to address students with content on their own skill level

2.3 Adaptability

The current adaptability of the game to the student is based mostly on the landscape line and objects that are drawn and the conflicts that have occurred before. When the student finishes the landscape line and Haas starts walking, certain conflicts are loaded based on a number of parameters such as the characters that are currently walking along with Haas, conflicts that have already occurred before, the shape of the landscape the student has drawn, the objects that the player has drawn between conflicts, etcetera.

For instance, when a player draws a lot of clouds in the air, the game might trigger the raining conflict, in which the student has to draw an umbrella for the main character. This kind of student initiated interaction has a great effect on the immersion of the game, though the educational effect can also be greatly enhanced when not only the game engine reacts on the players actions but the educational content as well.

In this part the ITS comes in, combined with a structured new way of creating content for the game using strict guidelines and wordlists indexed on difficulty and theme. An ITS for this game can truly enhance the adaptability to a higher level and create an even more effective educational game.

CHAPTER 3

Background research

This chapter elaborates on the literature research done to get a better understanding of what intelligent tutor systems are, what their features and components are, how we can dynamically adjust the difficulty in such a system and how these systems are able to improve the educational effect of a game.

3.1 Intelligent Tutoring Systems

A concise definition of an intelligent tutor system by Wenger and Ohlsson tells us that "Intelligent Tutoring Systems (ITSs) are computer-based instructional systems with models of instructional content that specify what to teach, and teaching strategies that specify how to teach" [22] [13]. If we want to reason about an ITS we have to separate several abstract components. First the instructional system, which is the actual system that gathers data, processes it and presents its results to the user.

Secondly, an ITS has a model of the instructional content, which is the content that we are trying to teach. This could be virtually anything. In a simulation environment, such as a helicopter pilot training program[1], this instructional content can be a set of rules and instructions that have to be followed to perform a flight correctly, whereas in a vocabulary extending game context such as Mijn naam is Haas this could be the target words that are meant to be learned.

The last part of an ITS as defined by Wenger and Ohlsson consists of the teaching strategies that specify how the instructional content will be teached to the student. This abstract action really varies per implementation, for instance using adapted instructions after a student error in a helicopter pilot training program or intelligently repeating target words at the most effective moments according to player skill in a vocabulary extending game such as Mijn naam is Haas.

For a more elaborate explanation of how we can use instructional systems to implement teaching strategies that "teach" instructional content, Fabio N. Akhras and John A. Self describe that "In ITSs, the knowledge to be learned is pre-specified in the system and knowing, according to the ITS view, is to possess this objective knowledge that is transferred to the learner during the instructional process by means of communication or problem solving. As a consequence, instruction is previously designed to identify and define the instructional methods that will help the learners in acquiring the intended knowledge" [1].

This citation identifies some key distinctions that have a direct correspondence to some of the key points in the developed MniH ITS. By following this citation and matching each component to the example of the MniH ITS we find that the pre-specified knowledge to be learned is the vocabulary that is embedded in the game's content base. The objective knowledge that is transferred during the instructional process consist of the contextualized words offered through the dialog and the instructional process consists of drawing objects according to the given instructions. The acquirement of the intended knowledge, that is a full understanding of all target words that were offered during the game and flagged as "completed" in the progress screen will be evaluated in chapter 8.

[1] AIS-IFT: Intelligent Tutoring System for Helicopter pilots: `http://www.stottlerhenke.com/solutions/training/ais_ift_its.htm`

To clarify that ITSs are not (yet) complete substitutes of conventional tutoring processes, Fabio N. Akhras and John A. Self explain that "ITSs are, in fact, just a particular kind of intelligent system to support learning whose components reflect the values of the particular view that ITSs emphasize in regard to then [SIC] nature of knowledge, learning and teaching, which have led to an architecture that focuses on representing the knowledge to be learned (domain model), inferring the learner's knowledge (learner model), and planning instructional steps to the learner (teaching model)" [1]. This shows that an ITS supports learning rather than replacing it, in a specific view that the ITS emphasizes on. Though the MniH game was never meant to fully replace the conventional vocabulary extension methods, it is seen as a worthwhile supplement to further enrich and broaden the student's word knowledge.

3.2 Abstract model of ITS

Specified in a previous citation, ITSs are structured in an architecture that separates and represents three different abstract knowledge models. Taylor elaborates on these models: "In order to provide ... adaptability to the student an ITS makes use of three knowledge models. These are the tutoring model, which regulates the instructional interactions with the student, the domain model, which contains the knowledge about the domain to be taught, and the learner model, which represents any information about the student. Adding the fourth component, the user interface, completes an ITS" [20] as cited by [4]. We can now construct a full ITS architecture example that fits the aforementioned citation and gives us a rough start for the MniH ITS architecture:

Tutoring model Contains all of the instructional tutoring interactions with the student, in our case the instructions to solve a problem through drawing objects and supportive feedback given based on right or wrong solutions.

Domain model Contains the knowledge about the domain to be taught, in our case the educational content delivered by educational experts who construct important target word lists categorized in themes, difficulty and priority.

Learner model Contains the information about the student, in our case a detailed report of all the interactions the student performs, the acquired scores, the amount of times contextualizing dialogs were played etc.

User interface The representation of the system to the users, in our case the serious game that interacts with the students and the teachers/parents backend that is used to give insight to the progression and skill level of the student within the game.

3.2.1 Domain modeling

When modeling domain knowledge, we have to take special care to how this knowledge will be used, according to LeLouche: "When modelling the knowledge of a domain, it is important to be aware of how this knowledge is going to be used by the system or by the users. In most systems, the knowledge is simply used by the system to make computations or retrieve information from that knowledge, as is the case in expert systems or, more generally, in knowledge-based systems" [6]. Even though the use case described by LeLouche is concerned with mathematical formulae used in cost engineering, the important remark is made that most of the time, knowledge is used in procedures to retrieve information. In our case, the knowledge is used to retrieve content for the most suitable target words that are available in the knowledge base.

When modeling pedagogical concepts into a knowledge domain, LeLouche [6] describes a technique using basic concepts inspired by entity relationship models and semantic networks to structure any domain knowledge, in our case vocabulary knowledge:

Basic concepts

The basic elements, involved in vocabulary teaching, show the notion of the concept.

Target word Word that is meant to be taught.

Definition The meaning of a word that is meant to be taught.

Description A sentence that explains something about a target word.

Derived concepts

Contextualizing sentence A sentence that gives a description of a certain target word

Dialog sentence A sentence without target words used to create a dialog

Sentence Sequence of words used in human language.

Relations between concepts

Theme a group of words that are related to a certain topic.

Network words words that have a direct relation to a target word that can be used to describe target words.

Passive word knowledge A student is able to recognize the word based on sound and / or shape.

Active word knowledge A student is able to use the word correctly in verbal utterances.

This technique gives a good structured overview of the domain that we are trying to comprehend and model into a knowledge domain.

3.2.2 Tests of intelligence

While modeling our system, we have to ensure that it still fits the ITS classification, for it has to be intelligent in some way. Angelides and Paul have devised three tests of intelligence that can be used to check whether a tutoring system classifies as "intelligent" [2]. First, the system must have enough knowledge of the subject matter to solve problems in the domain of application. Second, the system must be able to deduce a user's / learner's approximation of the domain knowledge. Third, the tutorial strategy must allow the system to implement strategies that reduce the skill difference between the expert and the student performance.

Keeping these three tests in mind, we can effectively create a requirements list of features the system must have to properly and effectively function while maintaining the classification of an "intelligent" tutoring system. In later chapters, the design of these key features will be described, and an evaluation of the three tests of intelligence will be given in chapter 8.

3.2.3 Student-centered tutoring

To provide adequate tutoring focused on the needs of students, or student-centered tutoring, Taylor describes an important requirement: "An ITS employs the knowledge of its three models to provide student centered tutoring, i.e. the adaptation of the tutoring process to the students needs and preferences. In order to provide this adaptability for a student, an ITS should apply suitable teaching strategies and presentations for each subject matter unit as needed, choosing the form that is most beneficial to the student for a particular instructional situation" [20] as cited by [7]. In other words, the maximum efficiency of an ITS is achieved when the subject matter is specifically tailored to the students instructional situation.

This is why we put much effort into creating separate teaching strategies and presentations for students on different levels (situations) in the MniH ITS. By creating a wide range of difficulty in the content and correctly annotating these difficulties, we are able to select the best content fit for a specific student, based on skills and previous play sessions.

3.3 Pedagogy and performance-oriented systems

When overviewing all possible ITSs, Murray roughly divides all systems in 2 distinct categories: "Early ITS authoring systems fell into two broad categories: those geared toward device simulation and embodying a "learning environments" instructional metaphor, and those based on a traditional curriculum (or courseware) metaphor. Even though some recent systems combine aspects of both perspectives, the majority of authoring tools fall similarly into two broad categories: pedagogy-oriented and performance-oriented. Pedagogy-oriented systems focus on how to sequence and teach relatively canned content. Performance-oriented systems focus on providing rich learning environments in which students can learn skills by practicing them and receiving feedback" [8]. When comparing these categories against the ITS that we are trying to create, we can safely decide that the MniH ITS is a pedagogy-oriented system, focused on how to teach canned content. Though fine motor skills and hand eye coordination are trained in the game as well, this is not the main focus of the ITS. However, while teaching the canned content the system does give constant feedback on the students interactions in a rich visual contextualizing environment, which leaves a hint of the performance-oriented category in our design.

When we decide to categorize the MniH ITS as a pedagogy-oriented system, there are some risks with respect to the instructional nature: "Proponents of constructivist learning theories (Jonassen & Reeves 1996) often criticize pedagogy-oriented tutors and the instructional design theories behind them as being too 'instructivist.' Such critics contend that these systems ignore important aspects of learning such as intrinsic motivation, context realism, common misconceptions, and social learning contexts. Actually these factors are acknowledged by most instructional design theorists (Merrill 1983, Gagne 1985, Reigeluth 1983), but are either seen as not being as important or as being too complex or incompletely understood to incorporate into instructional systems" [8]. However, the gameplay of Mijn naam is Haas pushes motivation that comes from inside an individual rather than from external rewards by immersing students in a fictional world and letting them explore possibilities freely.

3.4 Knowledge authoring and management

An important part of an ITS is the ability to add, modify and annotate knowledge domain content used by the system. Murray expresses some desired features in such an authoring system to simplify input: "ITSs are elaborate systems and authoring them involves managing a large amount of complex information. A number of common user interface techniques are used by various authoring systems to assist with knowledge management and organization. Simplifying input through the use of templates, data entry forms, and pop-up menus is quite common. Whenever the range of possible input values can be limited to a finite set, tools should be provided to allow authors to choose rather than type" [8].

Though not an internal part of my internship, MniH is working on a so-called "script and scenario tool" that is basically the authoring and management tool for the ITS game content. Keeping the aforementioned in mind, this tool is scheduled to be as simplified as possible, offering authors as much pre-defined simple options as possible. With this tool, an effort is also made to safeguard the complex requirements for the educational content, by implementing a number of checks before content is approved to be used in game.

3.4.1 Difficulty progression mechanics

An important part of an ITS, according to the three tests of intelligence, that *the tutorial strategy must allow the system to implement strategies that reduce the skill difference between the expert and the student performance*. In an article about the design of the educational game "JUMP", Rothschild explains how they have implemented this functionality.

"The predefined learning objectives for JUMP are to increase the players word knowledge by exposing him or her to approximately 500 vocabulary words over the course of 10 levels and to improve player use of word learning strategies by providing instruction and practice activities that target contextual and affix analysis. From the perspective of the player, the achievement of learning outcomes will be apparent as the player is able to progress from level to level, meeting the criteria required for game progression-criteria that are instructionally based. In addition to the indications of progressions in-game, JUMP features a built-in assessment database that invisibly tracks each players performance and dynamically offers review material specifically tailored to each players needs. The back end database tracks when and how many times players are exposed to specific words, their performance answering assessment questions, and the number of exposures players needed before they retained the reading strategy or word definition. Data tracked is cumulative and can be viewed for each level. This information is available through an in-game menu" [17].

In this system, student statistics are tracked "invisibly" (probably with a digital message containing game statistics that is automatically sent through the internet to an application programming interface of the game's server) and then used to draw conclusions on the progression of the student. Based upon these conclusions, the system assumes that after a number of word exposures the *skill difference between the expert and the student is reduced* and moves on to teach the next part of the knowledge domain. A similar assumption based system will be used in the MniH ITS, explained in chapter 5.

3.5 Dynamic difficulty adjustment

In addition to difficulty progression mentioned in the previous section, it is possible to implement dynamic difficulty adjustment (DDA). This real-time process uses player statistics to dynamically alter the difficulty of the game, based on the skills and abilities of the player. This is a great tool to enable the ITS to not only increase the difficulty of the game as the student progresses through the knowledge domain, but also decrease the difficulty of the game if it is too hard for the current situation of the student. An excellent example of a DDA system is given in an article about HAMLET, a DDA system implemented in the "Half-Life" video game by Valve:

"The HAMLET system is primarily a set of libraries embedded in the Half Life game engine. This includes functions for monitoring game statistics according to predefined metrics, defining adjustment actions and policies, executing those actions and policies, displaying data and system control settings and generating play session traces.

As the player moves throughout the game world, Hamlet uses statistical metrics to monitor incoming game data. Over time, Hamlet estimates the player's future state from this data. When an undesirable but avoidable state is predicted, the system intervenes and adjusts game settings as necessary" [3].

Though Half-Life is a non-educational, first person shooter game and has little resemblance with MniH, the HAMLET DDA system describes some key points to dynamically adjust difficulties in videogames in general and has proven to be a valuable source of information to implement DDA into the MniH ITS.

3.5.1 Assessment of flailing

The general goal of the HAMLET as described by Hunicke: "Basically, we are trying to predict when the player is 'flailing': repeatedly edging towards a state where her current means can no longer accomplish necessary and immediate ends. When we detect flailing behavior, we want to intervene by helping the player progress through the game" [3].

Detecting flailing behavior is essential for our ITS to be successful. Because students start somewhere in the middle of the difficulty scale, it is quite possible that the student performs worse than expected. When this occurs, the system should react as fast as possible by lowering the difficulty and stimulating the progression of the student on his own level.

3.5.2 Adjustment

Whenever flailing is detected, the system should somehow respond to this problem. in HAMLET, policies are used to determine what to do when a player fails to accomplish his tasks: "We are currently experimenting with a number of adjustment protocols in the Hamlet system. Adjustment actions, when combined with cost estimations, will form adjustment policies. What follows is a description of action, cost evaluation and policy designs, and an example illustration of the final goals for player-system interaction.

When completed, Hamlet will support two types of adjustment actions. *Reactive* actions will adjust elements that are in play or 'on stage' (i.e. entities that have noticed the player and are attacking). This includes directly manipulating the accuracy or damage of attacks, strength of weapons, level of health, and so on. *Proactive* actions will adjust 'off stage' elements (i.e. entities that are spawned but inactive or waiting to spawn). This includes changes to the type, spawning order, health, accuracy, damage and packing properties of entities that have yet to appear on screen" [3].

In this citation a clear distinction is made between two types of adjustment actions: Reactive and Proactive. When developing a vision for the MniH ITS we have thought of both adjustments. An example of reactive adjustment could be easing the solution object match to increase the chance the student draws the right object, effectively *manipulating the accuracy* of the user's actions.

However, in the MniH ITS we have chosen for proactive adjustment, by specifically matching the next conflict situation to the current level of the student. This design choice avoids confusion between students who might notice the reactive adjustment of the object match more than the proactive stage selection.

3.5.3 Comfort & Discomfort zone

To create an efficient learning situation within the game we would like to make sure students are constantly experiencing situations that are not too hard and not too easy. Hunicke calls this area between too hard and too easy the "comfort zone": "Hamlet can combine these actions and cost calculations to create 'modes of control' - overall adjustment policies that control the supply and demand of goods according to overall player experience goals. Two examples follow.

Comfort Zone: This policy will keep players within a mean range of health points over time. Entities will be tuned to shoot less often and less accurately, and health is readily available. The goal is to keep the player at about 50% health, occasionally dipping near 25% or cresting to 75%. Trial and error are important in this policy - enemies will be tuned only if they repeatedly overwhelm the player. Much like a benevolent babysitter, it intervenes frequently, but leaves room for mistakes. Overall, the policy will be characterized by steady demand and predictable supply. Discomfort Zone: This policy is designed for more experienced players. The entities in a DZ game are increasingly accurate, ammo and health relatively scarce. The goal here is to keep the player 'on the edge of her seat' - constantly on the alert for enemies, and fighting back from 15 or 20% health most of the time" [3].

Where first person shooters and other non educational games have the advantage of letting the player choose how hard the game should initially be, the MniH game should always adjust itself to one optimal situation. The so-called comfort zone for MniH is calculated based on the student's history of previous scores. This means that if the student is scoring average scores in the last played sessions, the user is in the comfort zone and we can slightly increase the difficulty to make sure the student progresses through the content. However, if a player is scoring poorly or above average in the last sessions, we should change the difficulty accordingly to get the student back in his/her comfort zone. More information on this algorithm can be found in chapter 6.

3.6 Vocabulary expansion

Because the main goal of the ITS is to develop the vocabulary of students at a young age and a lot of educational terminology is used throughout the rest of this thesis, I will highlight some of the educational literature studies that were used as reference material while engineering the MniH ITS. These literature studies are written in dutch by the "Expertisecentrum Nederlands" (EN)[2].

3.6.1 Principles

According to the EN, three basic principles are essential during vocabulary development for children at a young age: labeling, categorization and word network creation [12]. The first principle, *labeling* is the act of coupling the order of vocal sounds with the name of objects in the vicinity of the student. These labels are learned during the child's first year by interaction with adults.

The second principle, *categorization*, is one of the aspects of concept development that is concerned with classes of objects, symbols and events. Specifically, categorization is the ability of children to group similar words and to get a notion of generalization.

The third principle called *word network creation* means that children are able to see the connection between words on both semantic and syntactic level. Also, during this stage children get a notion of synonyms, functional relationships, and category specifications. These word networks are extremely important to understand and describe the meaning of new words.

This is why the content writing guidelines of MniH are geared towards target word contextualization using network words. By describing and visualizing a target word with (easier) words that have a close relation to the target word the student can first build the word network and then create active word knowledge about the target word.

3.6.2 Expansion Stimulation

To practically stimulate the vocabulary expansion, there are several methods, commonly used in the educational system to provide *qualitative* stimulation. Though it is important to offer language in large *quantities*, the content should be rich and challenging and may even contain words that are a little bit too hard for students.

These proven methods give a good starting point to develop an educational game with the same goal. As researched by the EN, the three important vocabulary stimulating methods are learning words through direct instructions of word meanings, learning through verbal context (e.g. picture books) and a combination of instruction and contextual learning.

Both strategies have a positive effect in different directions: direct instructions for learning word meanings and contextualized learning to improve the use of words in context. However, there is not one strategy that is preferred over the other so the combination is most effective according to Wood [23].

[2]The EN is a Dutch organization that translates scientific insights into practical solutions for educational purposes `http://www.expertisecentrumnederlands.nl/`

In addition to the rough difference in methodology, the EN suggests the following practical examples to be most effective to develop students vocabulary: reading out picture books, deriving word meaning by means of word analysis, using verbal and nonverbal context, writing a word in sentence and general repetition [12].

3.6.3 4-takt vocabulary teaching method

The creation of scientifically substantiated educational content for the MniH game can be achieved by following some guidelines provided by the written literature. One of these methods is the "4-takt" system approach [21] by Marianne Verhallen in which the educational process of learning vocabulary is split up in 4 separate steps:

Prepare The first step is activating the initial knowledge about the subject.

Semantize The second step is explicitly explaining the meaning of a word by explaining, expanding and visualizing the word network.

Consolidate The third step is practicing the meaning of the word by giving an instruction and repeating the target word.

Test The final step is validating whether the knowledge is transferred correctly in a playful manner.

These guidelines have proven to be a great source of information while trying to mold the available content into a teaching framework that is proven to be efficient and worthwhile. A brief overview of the MniH content writing guidelines using the 4-takt system can be found in chapter 5 section 5.4.

3.6.4 Feedback

Another crucial element of teaching any form is feedback. As an integral part of the instructional process, feedback can be described as all the communication used to supply a student with the validation of an answer. This communication has many forms, including direct corrections, further explanation, praising, punishing etc. However, according to the EN's literature research on feedback [11] not all feedback has a positive effect on the learning process.

First, feedback must be given in an understandable and relevant context. Second, feedback should be relevantly informative rather than just labeling an answer right or wrong. Finally, there should be no *presearch availability* in feedback, which means that students can adopt answers from feedback without thinking about it for themselves. The positive effect of feedback can only be achieved when students actively think about the problem and its answer.

As in any form of teaching, feedback is very important in language development. Especially feedback of adults on expressions by children have a positive effect on the development. Even negative feedback is essential for language development, for example providing correctional information on certain grammar that is incorrect.

In addition to verbal feedback from a tutor, be it a real life person or a virtual tutor, visual and auditive feedback is even more relevant for serious games. By providing non-verbal explanations and contextualizations an even more comprehensive associational network can be built to support the word knowledge. Also, when using a virtual tutor, facial expressions could add an extra understandable layer to the existing verbal dialog.

These observations are of great use while developing content for an educational game, clearly describing how content should be verbalized and visualized. Though not all of these points are implemented in serious games, these ideas have a big potential to even better supplement common teaching methods .

3.6.5 Word selection from word lists

To create an optimal range of target words that should be teached within a serious game, several word lists can be used that provide important words to learn, tagged with a difficulty value. During the MniH educational research two word lists were used as inspiration for the educational context, though the resulting official wordlist was heavily post processed by educational experts to categorize and prioritize the words before using the list as a rough sketch for the game's content.

After selecting the lists, target words that are not too hard and not too easy had to be selected as inspiration for the content that fitted the most suitable educational context. Two rules of thumb were defined by Kienstra about the ratio of new words in a text. First, texts should consist for 95% of words that students know already, which means that for every 20 words in a text 1 word may be unknown to the student. Second, a whole text should not contain more than 40 new words [5].

The second problem of compositing suitable content is that the difficulty of words is not easy to index and depends on a lot of factors. The most important grading factor for choosing a word should be the usefulness of a word, according to Schroten & Vermeer [19].

3.6.6 Word list difficulty

Available word lists such as the word list by Schaerlaekens [18] and the word list by Schrooten & Vermeer [19] offer thousands of dutch words in table form sorted by alphabet. For each word, the percentage of students who have an active knowledge of this word per group is given. This gives an insight about the necessity to learn certain words, and can be related to the difficulty or rarity of the word.

The difficulty of a word depends on three factors according to Nation: foreknowledge and native language, teaching method and the intrinsic difficulty of the word based on form and kind [10]. Of course, some kind of words are harder than others, for instance abstract and functional words are harder to explain than concrete words.

As mentioned before, repetition is also an important part of the learning process, for all new words are too difficult to learn at once. Robbins & Ehri [15] concluded in their research that while reading out a picture book, words have to occur at least 4 times in the story to be remembered properly. Unfortunately, no hard guarantee is given for this learning process, though a significant number of students remembered more words that occurred 4 times.

Reflecting on the MniH ITS, we have to presume that repeating a target word at least 4 times and thoroughly contextualizing a target word with network words has the best possible educational effect on students, closely resembling the reading out a story. We believe that because students are interactively involved in the story, the results will be even better when playing the game than just passively listening to a story.

CHAPTER 4

System design
Establishing a vision and methods

In this chapter, the initial design and vision of the system will be discussed by getting a clear idea of what the goals and components should be and what methods will be used to achieve these goals. Also, I will define the scope of my part of the project by analyzing the amount of work that has to be done to complete the ITS and to determine what part can be done within the timespan of my internship.

Another important part of the initial design was the co-operation with the rest of the creative team of MniH. Many requirements have to be elicited, because almost all of the active disciplines are in some way involved in the new ITS. Communication between the diverse kinds of expertise has proved to be an important factor in the ITS design process, because every stakeholder that is involved in the process has their own needs and interests.

4.1 Vision on MniH ITS

In this section I will articulate the vision of the ITS in my own words. Based on the available documentation and early discussions with the lead developer, this section was used as a verification on my briefing on the ITS development process. This vision is defined by reproducing the goals of the system in my own words and explaining the properties of the separate individual components of the ITS that have to be engineered and developed.

4.1.1 System goals

The first main goal of the ITS is to add a measurable and transparent educational meaning to the MniH game. By implementing such a system the MniH game evolves into a more durable product that can be pushed onto the educational market. In the educational business, it is very important for teachers and parents to get insight into the progression of students, which is one of the key stones in the development of the ITS system.

The second main goal of the ITS is to provide a large scale of content that can be selected by teachers to enable them to match the game's themes to the current theme used in the conventional education. Also, the available content should have a difficulty scale to broaden the target audience, to allow teachers to set the preferred difficulty of individual students and to enable the game engine to tailor the difficulty of the content to the skills of the students.

Another important goal of this project is that the current stand alone version of the game should be redesigned to work in a client-server model. By pushing most of the ITS logic to the server side, we are able to constantly fine-tune and improve the inner workings and maintain full control of all user data.

To realize the aforementioned main goals, an ITS must be designed and engineered that must be able to gather student information, process this information using computational models verified by educational institutions and present the system to the users and teachers in a suitable way, using the game as a front end for students and a web based interface for teachers and parents to view the progression.

Because content writers should be able to constantly add and improve the game's content assets, another interface should be constructed that can be used to efficiently expand the games content. The ability to annotate the educational material with difficulties is extremely

important to provide the system with a rich domain knowledge base used to effectively offer content tailored to the students.

4.1.2 System components

The ITS is a complex system that consists of multiple components that have to be either integrated in several existing components or developed from the ground up. Each of these components is carefully specified and described early on, though the eventual architecture of these components will be devised in a later stage. The following subsections describe the meaning and goal of each of the individual components that are required for the ITS to function properly.

Game front end

The game front end, as described in chapter 2 is an existing component that has to be enhanced with statistics gathering and scoring functionalities, the ability to identify the user and the ability to send the user statistics to the game server. To identify the statistics that have to be gathered we have to construct a learner model that uses this data to draw conclusions about the skills of the student.

Because the interaction with the user is solely based on drawing interactions, it is very hard to draw definitive conclusions on the skills of the student. Therefore we have to carefully select the possible statistics such as duration, match scores and reaction times to decide whether or not a player is doing well. To react on these statistics in a robust way, we try to use the average statistics of as much students as possible, which is one of the few certain factors for the scoring calculations.

The game user interface visual representation isn't meant to change with the implementation of the ITS. Except for a student identification screen, the ITS should be as invisible to the student as possible, to avoid any interruptions to the immersion of the player. The statistics and scores that have been acquired will all be silently sent to the server and are only displayed on implicit requests of teachers and parents.

Teacher console

The teacher console is a web front end that offers direct insight into the progression of the students to both teachers and parents. In this web application teachers are able to see the target word lists that correspond to the chosen difficulty levels and the target words that have occurred during individual student's sessions. In addition, several class views will be constructed that give a clear overview of the whole class's progression with averaged student scores and statistical information about the words that occurred during the game sessions.

The exact specifications, requirements engineering and design of the teacher front end is done by the companies interaction designer. By performing requirements elicitation, prototyping, discussions with peer groups and several pen-and-paper tests, the exact design of the teacher console will be defined. The implementation of this system is depending on the architecture of the database and will be kept in mind while designing the database. Close communication with the interface designer was maintained during the process to reassure that all necessary functionality was implemented, because all the required statistics must be stored and processed in the database back end.

Content supplier console

The content supplier console, also known internally as the "script and scenario tool" is a tool for content suppliers providing an efficient and resourceful interface to deliver input and maintain the game's (educational) content. The functional design, created by the interaction designer, was in full development during my internship. Therefore, to reassure that

the engineered models correspond to the new methods of inserting content into the system's knowledge base, close communication with the development team and content writing team was required.

With this tool, content writers can insert the future ITS enabled content using the new educational framework (described in section 5.2). All future content will be written according to the new educational writing guidelines and can be inserted into the game framework in a quick and structured way. This provides the game with rich educational content without impeding the writers creative process.

Back end content API

A context and student skills aware back end content API should be designed to provide tailored educational content on game requests, because the content of the game is structured in a whole new way and should now be delivered to the game dynamically in a client-server oriented structure. By keeping all game content on the server side the company has full control over the game's content with the possibility of adding and adjusting content in real time.

The main goal of this back end content API will be to process incoming game client content requests as quickly as possible and respond with the content that is most suitable for the student's current skills and progression. This API will implement several conceptual models, described in later chapters, to decide what target words have a high priority for the current student and which target words are closest to the determined preferences of the student in the current situation.

Back end storage API

Next to the available service call for loading content, the back end should be able to receive, store and process game client session statistics packages. These statistics, acquired through this form of data mining are essential to adjust the system's learner model and to get an insight into the progression and skill of the current student. Furthermore, these statistics are used by the teacher front end to show the statistical progress of the students in viewable user profiles.

After the statistics are received and stored, the computational model for the dynamic difficulty adjustment system is used to determine the preferred skill level of the next session, according to the achieved scores, previously achieved scores and the average score of all students in a recent timespan. This processing should be done as efficiently and quick as possible to ensure stability and robustness of the server during system usage peak times.

4.1.3 Reusability

In an ideal situation, the ITS should be a generalized, interchangeable component that can be reused in other serious games. However, in our case the ITS has a large impact on the way content is supplied for the game, so this part of the game is quite interweaved with the game's component, playing an essential role in the selection and priority system of the offered educational content.

However, the statistics storage component is quite reusable in future projects, in conjunction with the stats processing component and the teacher console. Though initially written for vocabulary expansion, this system provides a clear and structured flow of generating session statistics, sending these statistics to the server, storing these statistics in the database and processing these statistics to adjust the system's learner model, which could be reused in any form of tutoring system.

4.2 Internship scope

In my internship, I focused on the engineering of the ITS components, conceptual and computational models and the system's detailed architecture. For the remaining time of the internship after the engineering phase I started the implementation of the server side back end logic, according to the initial planning.

To define this scope, we made time calculations and strict planning for each of the individual tasks that had to be completed in order to engineer the whole ITS system. We decided to exclude the content writing tool from my internship to allow full focus on the systems engineering, because this component wasn't directly involved in the ITS logic.

4.3 Influence of multidisciplinary team on system design

During the internship, a lot of communication was necessary with a lot of the internal and external employees. In the starting period of the internship, I had created an internal document with questions for several different parts of the MniH team that would have to be answered before we could start creating the design and conceptual models. The list also provided some insight into the nature of the system for the employees that weren't up to date with the goals of the ITS.

In addition to the questions list, a lot of general communication with the different company groups was necessary for the development of the ITS. To give an overview of all of the different groups that have something to do with the ITS, I will shortly describe each of the groups and list some example questions that were asked to guide our engineering process in the right direction.

4.3.1 Educational experts

The educational expert played a major role in the development of the ITS, safeguarding the educational substantiation of the system's inner workings through experience and research. By providing feedback and evaluating the created models, we could use this specialist expert knowledge to create an advanced system for this learning domain.

A big influence of the educational experts were the teaching rules that had to be obeyed in order for the game to have any significant educational value. These kind of rules had a great impact on the conceptual models and some models are even based solely on these kind of expert rules. Some questions that were asked to the educational experts were "How do we teach children vocabulary", "How many times do we have to repeat information, and when" and "How do we define progression in the world of MniH".

4.3.2 Content writers

The content writers write content for the MniH game and had to radically adjust their writing to match the guidelines and frameworks that were written for the ITS enabled version of the game. We had to make sure that none of the devised models forced the content writers to alter their ways so much that the original feel of the game would be altered, because the style of writing is very important to the atmosphere, feel and gameplay of the game.

The biggest influence the content writers had on the system was the trade-off between how educationally correct the game should be versus how fun the game should be to play. Another important influence was the feedback and approval of the new way of structuring dialog (as described in section 5.2) which had a big impact on the amount of dialog that had to be written in order to broaden the difficulty range of the content.

An example of some questions that needed answering by the content writers were "What is the language difficulty in MniH, and on what content level should it be defined?" and

"How much time is needed to write dialogs in the new stage framework per stage and plugin?".

4.3.3 Interaction design

Interaction design had a big role in the development of the content management tool and the development of the teachers console. In order to create a database system that suited the required statistics and the teacher's needs, we had to closely communicate with the interaction designer that reported the needs of teachers.

Through peer group meetings and usability testing, the interaction designer had much insight into the audience that will be using the eventual serious game and has proven to be a very important resource of requirement information for the front end interaction of the ITS and the storage of statistics.

Questions asked to the interaction designer included "How do we create insight into the students progression?", "How do teachers want to use this product within the curriculum?" and "How much control should a teacher have over the ITS specific variables?".

4.3.4 Business logic / marketing

The business logic and marketing group maintains contact with elementary schools, financial investors, publishers and others to deploy the product on the educational market. This group decides over licenses, fees and separating game content in separate titles of the game that need to be published.

This group also has more insight in the future of the company and is concerned with internationalization, licensing and scaling of the products, which has a major influence on the implementation of the ITS. Several questions that had to be answered by the business logic and marketing employees are "Will all future titles only be ITS enabled?", "Will CDROM only titles still be created?" and "What is the functional difference between school and home copies?".

4.3.5 Developers

Developers are the employees that have a lot of knowledge of the current game engine and have a good insight into future technologies within the game. Therefore, this group provides a great resource of inspiration for the ITS development. Furthermore, it is important to keep in mind that the created architectures stay implementable within the given timespan, because the developers have to implement the final version of the ITS.

No explicit documented questions were asked to the developers, though this group was kept closely in the feedback loop to make sure that all architectural designs were implementable within the current technology and available timespan.

4.3.6 Management

The management main role in the ITS development was to keep track of the schedule that was created containing all the development milestones and components that had to be engineered and implemented. By keeping the management in the communications loop, they were able to keep track of the ITS development progress and compare the results with the schedule we created. Though no explicit questions were formulated in the questions list for the management, they have played an important part in safeguarding the timeframe and schedule of the development.

4.4 Research Methods

This section will give a brief overview of the research methods used to achieve the research goals of this project. By combining all of these methods, we are able to get a complete and thorough overview of the complex structure and stakeholder's needs of the MniH ITS. This overview is essential for several reasons. First, it is necessary to create an implementation that is educationally substantiated. Second, it is needed to make sure all the needs of its users are satisfied and finally, because the project should stay realizable within the given development timeframe.

4.4.1 Literature study

The first method of literature study is used to get a firm knowledge base on intelligent tutor systems by gathering previous research done throughout the world. Much research has already been done on this topic with promising outcomes, because tutor systems can be used in any virtually any tutoring situation and provides efficient supplementary methods to tutor content in an intelligent way,

By creating the design of our ITS based upon proven architectures and methods we make perfect use of previous research and gain some time to engineer advanced functionality on top of the provided frameworks. Some background research will be done on the educational content that is inherent to the system to provide a better view on the educational means and goals of the system, because a large part of the ITS inner workings cover a lot of domain specific knowledge,

4.4.2 Discussions with engineers

The engineering discussion brainstorms have proven to be the most important and essential research methods during this internship. These meetings, usually taking 1 to 2 hours, were scheduled in advance with a certain topic or problem that would be processed into a conceptual model. By preparing these meetings with initial ideas and suggestions, direct feedback of fellow engineers could be used to improve and mold the rough ideas into actual usable concepts.

After these sessions the notes made on the whiteboard and paper would be transcribed into a debriefing document which carefully wrapped up the created concepts and ideas. These documents could then be used to further evaluate and refine the models, as well as communicating the concepts and ideas to the rest of the team to check whether or not all the needs of the different stakeholders were still satisfied. Later on, these documents were used as blueprints to create the computational models and architecture that were used to implement the system.

4.4.3 Discussions with developers

Likewise, discussions with developers have been equally important to get an overview of the current state of the project. These meetings were much less formal and usually took less than an hour. The goal of these meetings in the start of the internship was to get acquainted with the game engine and code base, getting a clear view of the current technology and sharing ideas about future possibilities.

From an architectural point of view, it is very important to not blindly focus on the functionality of the system but to keep the developers and their needs in mind as well. It is important that all desired functionality is implemented in the architecture, but at the same time these architectures should be implementable within the given timespan. Though this not only implies a restriction on the systems complexity it also asks for creativity, which are both essential to the succeeding of the project.

4.4.4 Discussion with educational expert and expertise centre

The planned meetings involving the educational experts and the EN were planned far in advance to ensure the presence of several internal and external people that are involved in the educational part of the project. These meetings, ranging from 2 to 6 hours were carefully prepared and specifically aimed to elicit feedback and possible approval of the created models and structures.

By preparing these models in a readable form and explicitly explaining these through concrete examples, we were able to ask specific and direct questions about the assumed effect, validity and performance of the created system. The feedback that was generated during these meetings offered proper handles to improve and finalize the concepts before the actual computational models and architectures were devised.

4.4.5 Feedback through presentations

During the development of the ITS, I have held several presentations regarding the process of the ITS development. During these presentations, I have tried to give an overview of the current state of the project in an understandable manner. These presentations usually took place right after a team meeting, in the presence of the majority of the companies employees.

Preparing these presentations forced me to rationalize and structure my design decisions which was a very important step in the process of creating an understandable architecture. During and after the presentations, I have gathered a lot of feedback from the rest of the team about ITS functionality and remarks on the proposed architecture. This really put the architecture in perspective and allowed me to validate the effect of the system before actually implementing the system.

Another important positive effect of these presentations was progress notification of the ITS to the rest of the team. During these meetings, the team had a lot of freedom to react on the ITS and were able to express their needs and requirements by providing feedback and asking questions about system details.

CHAPTER 5

Conceptual Models
Difficulty, progression and scoring

The second big step in the process of realizing the MniH ITS was the definition of the conceptual models from the elicited information during the design phase. These conceptual models give a rough sketch of the tutoring mechanics of the ITS that covers the tutoring model, one of the three abstract models of an ITS, as described in chapter 3.

In this chapter, the tutoring model will be explained by demonstrating how the content will be ranged in difficulty, using a new stage framework that expands the difficulty range of the dialog by using the same situation for different target words. Also, the writing guidelines for the content writers according to the 4-takt educational model are described, as well as the content dispense rate, the progression tracking and solution scoring systems that have been thoroughly discussed and evaluated in a meeting with the EN.

These conceptual models provide the information needed to grasp the scope of the system. By defining and validating these functional tutoring strategies, we can cover the tutoring part of the requirements and eventually create an architecture that instantiates the models into an architecture that separates the conceptual models into implementable abstract parts.

5.1 Word difficulty levels

The target word list that is used as a source for the vocabulary domain knowledge consists of about 6000 words, each of which is labeled with a difficulty value. This difficulty value is made up of the percentage of teachers that think that that specific word is considered important to learn in the "groep 2" school class. These difficulty values range from 0 to 100 percent, as seen in figure 5.1. The words with value 90 through 100 are considered unanimous words, which means that these words are considered to be known by all students and can be used freely by the storywriters to create context for the target words. The words below value 30 are considered to be too insignificant to get used in the game. This leaves words with values between 30 and 90 to be selected as possible target words to use within a stage.

These possible words can be divided in 6 levels, which gives six distinct levels of words[1]. By using a small scale it is easier for a teacher to estimate whether the student should start on low, medium or high level target words. Internally, we use the larger scale of word difficulty percentages to add more variety in difficulty within each level. To get a global view of the progression of a student we create a *level progression value* for each of the available levels, so we can keep track of the student's progression within each separate difficulty level.

Initially, this level progression value is based on the number of "completed" words per target word difficulty. When the student "completed" a certain number of the target words with difficulty level 1, we presumed that the student was ready for more difficult target words and we could set the preferred level slightly higher. However, we found that this would take too long to quickly adapt to the users skill level and gives little control over lowering the difficulty.

This is why we used the acquired session scores to decide whether the students preferred word level increases or decreases. In this way, the ITS has more dynamic control over the difficulty of the game. The statistics screen can still show the number of words that is com-

[1] These six levels will also be used in the EN word test tool.

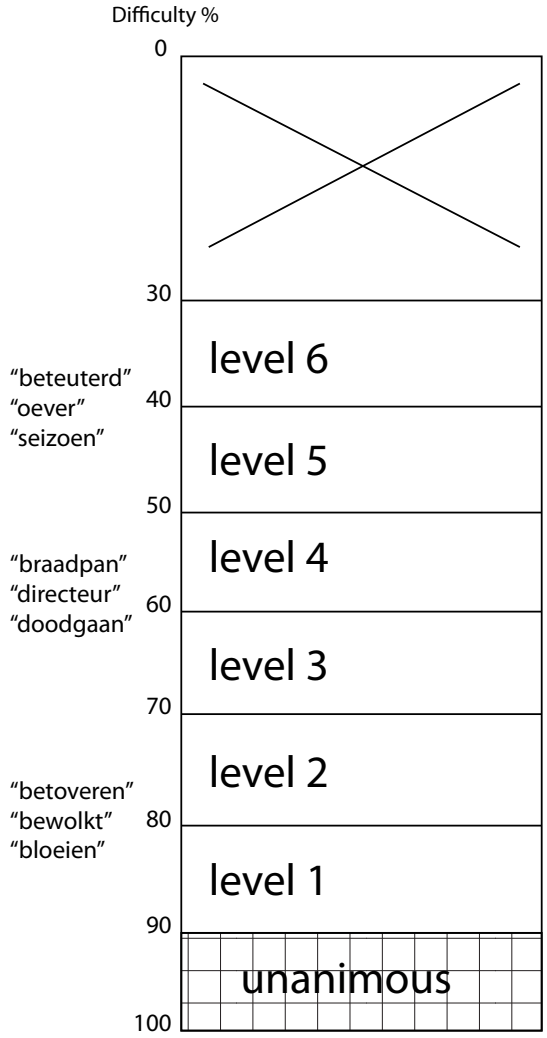

Figure 5.1: Word level diagram, showing words as examples for the word levels levels, the difficulty percentage on the y axis and a clear distinction of 6 word levels, unanimous words and words too insignificant to use within the game.

pleted and using lower priorities for completed words, the word preference automatically shifts towards the new words that haven't occurred before.

During the class registration process, the preferred word level can be preset by a teacher for individual students allowing skilled students to start with higher level target words and vice versa. We can show the teacher which words are associated with a certain level because all the target words are labeled with a difficulty.

Determining this initial word level could be automated using an external vocabulary testing application. This same application could also be used to measure the vocabulary skill of students after using the MniH game for a period of time[2].

5.2 Stage Framework

The stage framework diagram describes the proposed structure for the new ITS ready conflicts. A conflict is a generic scenario in which the main character faces a certain problem. The generic template of the scenario is described in the "base class". However, the exact content

[2]Such a word level test application currently in development at the EN which will also be used to measure the effect of MniH.

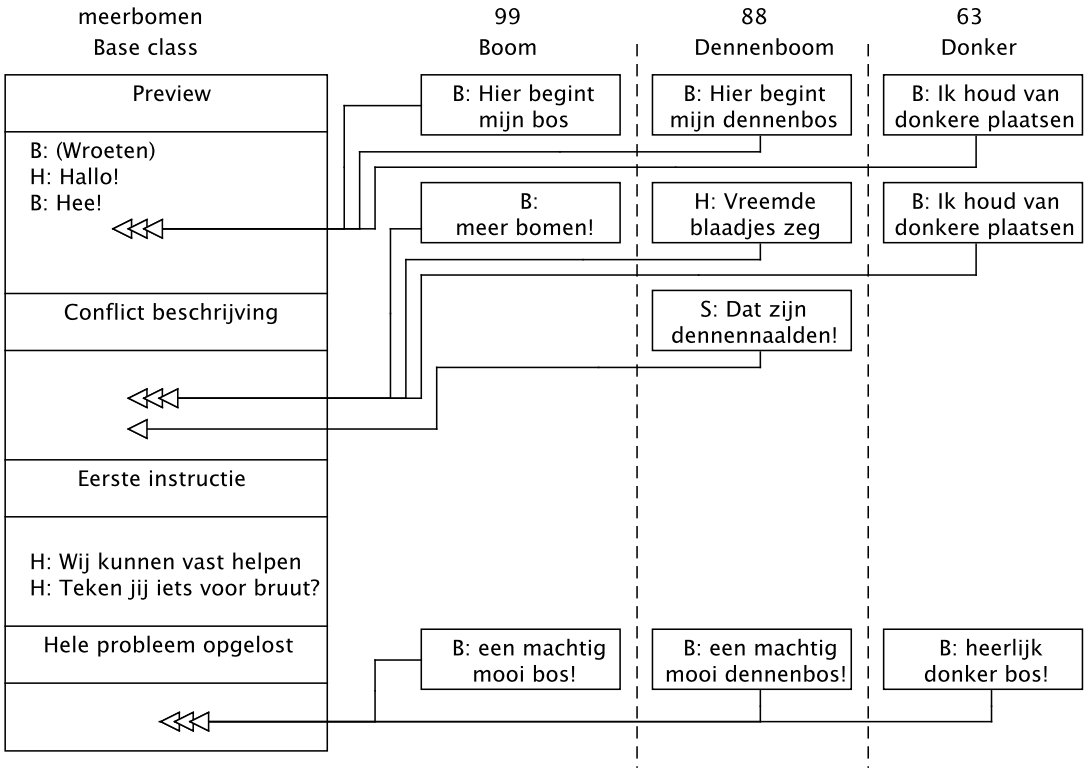

Figure 5.2: Stage Framework diagram, showing the base class on the left and three plugins on the right.

is described in the "plugins" that contain different target words of different difficulty levels. In this example, the base class describes the animation, the greeting and some instructions that are standard for this conflict, while the plugins "Boom", "Dennenboom" and "Donker" each contain the specific target word related dialog that plugs into specified locations in the base class.

In an actual occurrence of this conflict in the game, the preferred word level for that specific user is consulted to find out which of the plugins suits the user's level best. When the right plugin is chosen by the game, the base class is filled with the content from the plugin in this way filling empty spaces and overwriting dialog from the base class. Alternatively, the plugin can override the base template or provide *alternatives* for base template dialogs. This results in one linear set of dialog that is used within that specific conflict, matched to the user's level. In this way there is no need to duplicate conflicts when using different target words in the same conflict situation.

In a concrete example, when the framework shown in figure 5.2 is used and the preferred word level of the current user is 90, the plugin "Dennenboom" will be chosen (for its difficulty of 88 is closest to 90) and the content of that plugin will plug into the base class. This will result in the conflict seen in figure 5.3.

5.3 Difficulty and progression scenario

Using a specific example, this section describes the events that occur and the reasoning behind the steps that are taken to select content that is offered to a specific player. This specific player is Maartje, she is 5 years old and has been playing MniH for a while now. Her preferred word level is now 90, because of her acquired progression in the game. Maartjes teacher has set the theme of MniH to "tijd / seizoenen", for that is the current topic within the class.

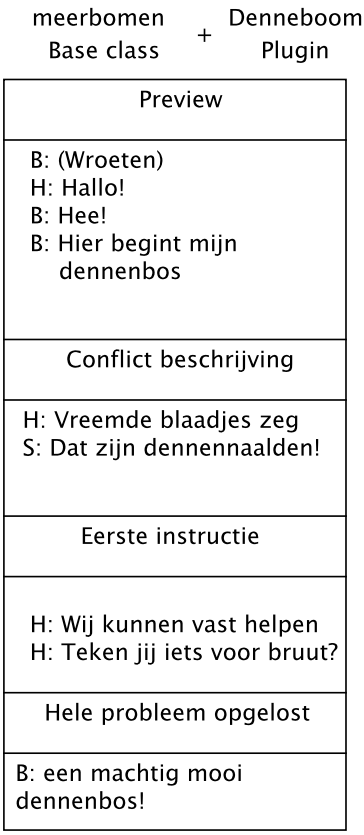

Figure 5.3: Example of the resulting dialog using the base class of "meerbomen" with plugin word "dennenboom".

5.3.1 Difficulty and Theme

When Maartje starts up the game, she is asked to draw a landscape for Haas. After doing so, the stages are selected that will take place during her play session. In this case, the first stage that is selected is "meerbomen", for its required landscape matches the landscape Maartje has drawn, it is labeled with the theme "tijd / seizoenen" and Maartje's preferred word level is within the word level range of this stage.

The stage "meerbomen" has four possible target words that can be learned within this stage. In this case, the word "dennenboom" is closest to the desired word level. Though the goal of this stage is to draw more trees, the dialog that is used within the stage now has focus on teaching the target word "dennenboom" by repeating this target word within different contexts within the different steps in the stage. However, this stage could very well be focused on the target word "donker" when Maartje is ready for more difficult words.

When the best matched target word is chosen by the system, the associated base class and plugin dialog is fetched and played within the game in which Maartje is asked to draw multiple pine trees.

The following two stages in Maartjes play session are chosen in a similar way though these are not the same stage or target words as the first stage. When the session is done, the statistics from the session are transferred to the server, which stores and processes the progression Maartje made.

5.3.2 Progression

The target word "dennenboom" is considered to be handled once and needs to be repeated some more in a following session, because Maartje has drawn the pine trees in the first stage

when asked, didn't draw too much wrong sized objects and didn't let the timer run out that automatically solves the stage.

One week later Maartje plays the game in the class again and the process of selecting the right stages starts over again. However, the system has registered that the word "dennenboom" has a high priority because it has to be repeated some more to achieve the best educational result. Therefore, a stage containing that target word is selected, preferably a different one than the last time. In this case, there is another stage that handles the target word "dennenboom" in a slightly harder context, the so-called "geenbomen" in which there is less visual context (no trees).

At the end of the session, the results are transferred to the server again. The target word "dennenboom" is considered learned and therefore the priority of the word is lowered, because Maartje has done well and completed the task at hand (draw pine trees). The next time the game is selecting a stage, other target words will be taken into account. When more target words are flagged as "learned", the preferred word level is changed accordingly, ensuring that the target words Maartje encounters in upcoming sessions increase in difficulty.

5.4 Writing guidelines according to the 4-takt method

During the writing process of the MniH game content, several guidelines have been posed to safeguard the educational value of the spoken dialog. By following these steps we can assure that target words are sufficiently contextualized and that students have more than enough opportunities to build a word network to get active word knowledge about the word we are trying to teach. The writing guidelines we have constructed are based on the 4-takt system, briefly described in section 3.6.3.

Each conflict that occurs in the game is separated in so-called conflict steps. These conflict steps consist of 15 possible parts of a conflict that each have their own role in the process of creating the context for the problem. Each conflict step has its own function in the dialog, for instance introducing the problem, introducing the involved actors, providing instructions or giving appropriate feedback to the student's actions. By creating a strict guide in which conflict step we place the important target word content we can effectively implement the 4-takt teaching method into the MniH serious game.

Prepare The preparation of a target word should take place in the conflict steps "preview" and "subject feedback before solution". By using simple unanimous words, the context for the target word should be created. When these context words are consistently picked from the current theme / category, the correct existing knowledge is "activated", which is crucial to support the semantic step in the process.

Semantize The semantic process takes place in the conflict explanation, where the meaning of the target word is explained through the elaboration of the problem. By creating extra dialog that describes and repeats the question in another context, such as another actor stating the problem again, the word network is expanded. The visual representation on screen should support the word network. For example, when asking for more trees, it is very helpful when there are already some trees present.

Consolidate / Test The main consolidation (or "word practicing") is done in the instructions conflict steps, and the complete, partial and wrong solution conflict steps. In the instruction steps a direct question is posed to the student who has to react to solve the problem. By repeating the target word multiple times in a short time, in the complete, partial or wrong solution feedback on the action, the learning effect is enhanced. In this way, the knowledge is tested in a playful manner without punishing the player for giving the wrong solution, rewarding the player when the solution is right with visual and audible cues.

5.5 Vocabulary tutoring by intelligently repeating target words

The hardest part of creating an educational game is to ensure that the offered knowledge, in this case target words, are actually "learned" by the student. To stimulate the learning effect we have to disclose the knowledge in a proven educational way while keeping the game playful and dynamic in order to keep the student motivated. To achieve this we have to safeguard some important content features that concern repetition, alternative contexts for target words in different situations etc.

The following repetition scheme can be used to effectively tutor the knowledge within the game while retaining the dynamic selection of game content that is true to the games nature.

When writing a stage that treats a certain target word, this word has to be repeated at least four times throughout the different steps stage. In addition to repeating the target word within one stage, the educational effect is maximized when the same target word is repeated about three times in the next session, preferably in a different context. To achieve this optimal learning result, it is important that the ITS stores information about the sessions and is able to respond to content requests with content that is tailored to the current progression of the student.

This results in the following structured flow within the game code, using the *priority* of each target word to decide which content is preferred for that specific student.

1. Register the session, store which stage was selected and what target words were used in the dialogs

2. At the server side, the stats are stored and the priorities for the target words are updated according to the resulting *solution score* (details in section 5.8).

3. When the next session is played, we select the stage that matches the highest priority target words, and prefer the stage that handles the high priority target words in another context.

4. If the student acquires a high solution score in the repetition of the stage, raising the *learning completion value* (details in section 5.8) high enough to label the target word as "completed", the priority of that target word is lowered automatically.

The end result of this scheme is that each target word is repeated over a certain number of sessions. After these repetitions the priority of a word is lowered (for it is considered "treated") to ensure that there is a bigger chance that new words are chosen. It is possible to view this information in the teacher console, because all the sessions and stages are logged.

5.6 Content dispense rate

The goal of the ITS will be to treat 6 target words in one session. One session will take about 10 to 15 minutes to complete. During testing, students will do 2 sessions per week. This means that 12 target words per week are treated, resulting in a total of 96 words treated, during 8 weeks of testing.

To achieve this rate and still repeat target words in different sessions, target words have to be repeated outside the conflict stage, which could be done by creating free drawing reactions and neutral stages for each target word that can be placed in any context, giving room for new target words in the conflict stages.

Though this behavior would be ideal, it is not considered in the current architecture. For now, words are always repeated at least once in the following session, no matter how good or bad the student performed. If the student performs averagely or better, the word is considered completed and the priority is lowered.

5.7 Progression tracking

When deciding what plugin target word to play in a certain dialog, there are a couple of factors that have to be taken into account when the response has to be personalized to the student's skills. Before we can personalize the query results, we have to store the statistics about the sessions played by the student to indicate what his or her level is.

For each student, each in-game encountered conflict and its contained target words are registered along with some scoring values. The first value is the target word occurrence count: amount of times this target word has occurred in the game. This value is used to provide some more information in the statistics view about the relation between the amount of times a word occurs and the acquired score.

When the stage is over, the target word has occurred in a contextualizing dialog sentence for at least 4 times, as prescribed by the writer guidelines. However, we would like to repeat this word again in the next session, which is usually played about one week later. To achieve this we use the amount of target word occurrences and the acquired scores of students to calculate the priority of a target word.

On the end of every conflict a solution score is calculated from the user's action and this score is registered in the database. By calculating this score between a minimum of 0 and a maximum of 1.5[3] based on the "correctness" of the answer we can test how well the meaning of the word was understood per repetition. Even when the maximum score of 1.5 is awarded the target word has to be repeated at least once more in a later session in order to achieve maximal educational effect. However, when the word is repeated and another good score is achieved, the target word is rendered "complete".

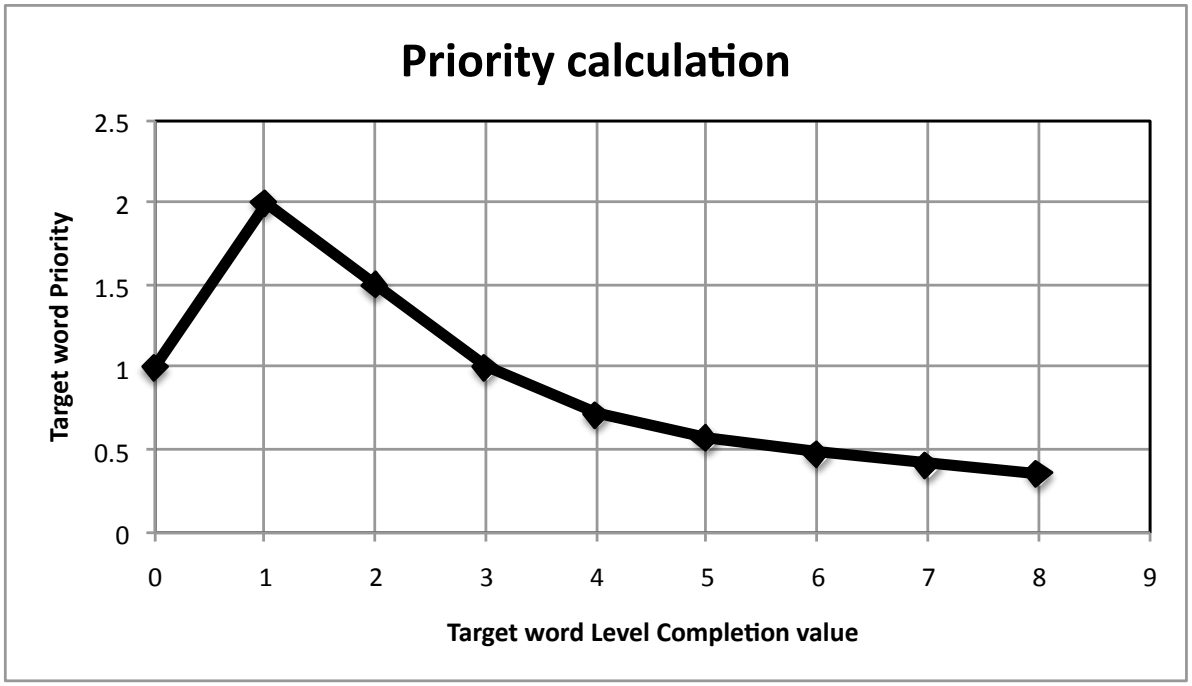

Figure 5.4: Target word priority calculation based on learning completion value which was later rewritten into more reliable priority calculation algorithm.

In the start of the project, we invented a learning completion value that was directly linked to the priority of the target word, as seen in figure 5.4. Every solution score achieved by the student would be added to the learning completion value which would then have effect on the priority of a target word. All learning completion values started at 0 and go up infinitely,

[3]Please note, all written values are indicator values and are subject to change when testing and fine-tuning the system.

because we couldn't completely exclude words of a certain difficulty level from occurring because they are needed in certain conflict constructions.

However, we found that the learning completion value wasn't able to give an accurate prediction of the target word priority. The algorithm used to calculate the priority of a target word was rewritten to use the amount of occurrences and the averaged solution score difference, as described in section 6.4.2 which gives a much more specific, reliable and adjustable prediction for target word priority.

5.8 Solution scoring system

The solution score is calculated in a similar fashion as the object match score in the current MniH game structure: the object match score is used in the game engine to determine what object is best matched to the line that is drawn using some metrics like line length and angle. To determine the solution score after finishing the conflict, we use some sort of metric to measure the correctness of the users actions while solving the problem. The measurable options that provide the most reliable conclusions are the following:

Instruction loop index This index counts the amount of times the instructions have started over because no right solution was drawn. If this index is very high we can indicate that the student might need some more repetition and therefore increase the learning completion value with a lower solution score.

Auto solve When the instruction loop index reaches a predefined maximum and the conflict has auto solve enabled, the conflict is automatically solved by the game. This means that the user has got all of the instructions multiple times and still hasn't drawn the correct object. Whenever this occurs we set the solution score to 0 to make sure these target words learning completion isn't raised and that the word is repeated after a couple of sessions.

Wrong objects count While the student is solving a problem, he or she might draw a number of wrong objects before the correct object is drawn. To punish the player for drawing incorrect objects, we can decrease the solution score with a certain amount for each wrong object, or subtract a fraction of the score for every 5 wrong objects.

Match score The match score is a game feature that checks the drawn line and matches it to the active objects that can be drawn at that moment based on line size, line angle, distances to actors. If a wrong object is drawn, the match score of that line could be used to define a conceptual "distance" to the match score of the right solution. This distance could then be translated in a penalty which is subtracted from the solution score or modify the weight of each individual wrong object in the wrong objects count penalty.

To calculate the resulting solution score we will have to create a weighting system for each of the metrics that makes sure the focus of the value is based on the right metric. To keep the most control over this score we could award 1.5 solution score and subtract penalties based on the aforementioned metrics. An alternative scheme would be to award points that add up to a solution score of 1.5, but this will very likely not result in expectable results. Generally, a student plays the game and completes the challenges, resulting in a predictable repetition scheme. When a student performs sub-par, we can create penalty points to increase the number of repetitions.

When the game session is done, the resulting solution scores for each target word should be stored in the database, along with other statistics about the played session such as the amount of target word occurrences etc.

CHAPTER 6

System architecture
Communication, processing and database storage

Introduction

The third and biggest step in the MniH ITS development process was to create an architecture that included all the created conceptual models and to abstract this architecture into several documented components that can eventually be implemented in a chosen technology.

This architecture gives a structured overview of all the client-server communication, statistics storage and processing algorithms and a comprehensive data model that is used to represent all the available domain model, tutoring model and learner model knowledge. The contents of this chapter consists of documented diagrams, detailed component descriptions and algorithms that can be used as clear programming guidelines when implementing the ITS.

Several algorithms explained in this chapter use mathematical formulas to calculate certain values. All these mathematical formulas including the algorithm variables, which will be fine-tuned during the testing and refining process, can be found in appendix B.

6.1 API / Backend Communication

In figure 6.1 the communication between the game client and the game server is shown according to the call that is made to the server, the parameter objects that are sent along with the call and the response objects that are sent back to the client. There are five separate server calls that can be made by the game to perform certain actions, retrieve content or store statistics.

The available calls are offered through a service gateway via Remote Procedure Calls (RPC). The service gateway technology used in MniH is AMFPHP[1], which allows for binary serialization of native types and objects to be sent to server side services. Calls are prepared with typed parameter objects, which are serialized and sent to the server.

The server receives these typed parameter objects, performs the appropriate queries to the database and constructs a new typed object containing the results. This result object is serialized and sent back to the game client, which receives the typed object after deserializing the result. The contents of this typed result object can then be used in the game's core.

More information on the system's interactions, detailed API backend scenarios describing call name, actor, summary, scenario steps, assumptions, exceptions and results of each scenario can be found in appendix A.

[1] AMFPHP, `http://www.amfphp.org/`

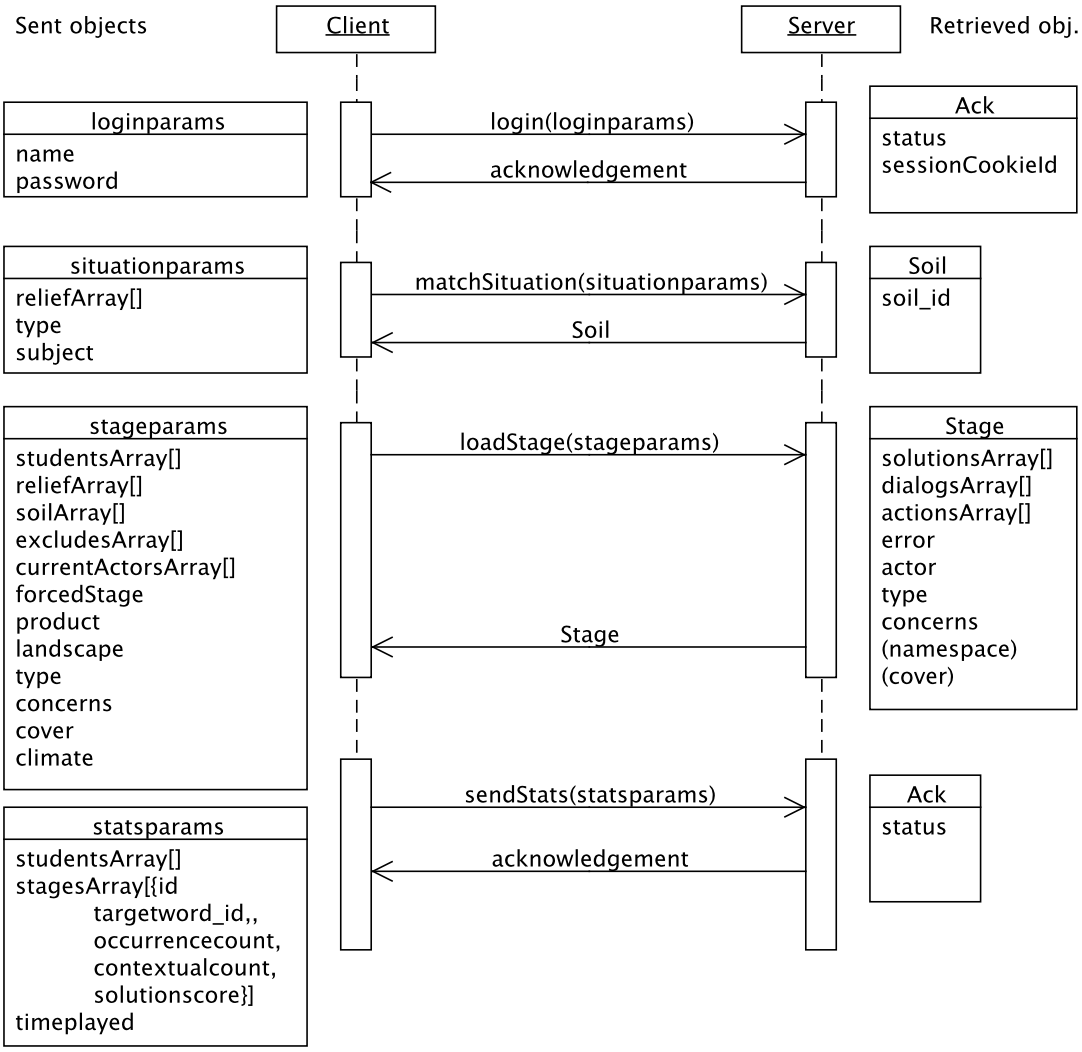

Figure 6.1: Communication between client and server, including objects sent and received.

6.2 Statistics display

The goal of gathering, sending, storing and processing the statistics is to keep an accurate user profile in the system's learner model which in turn allows the tutoring model to provide appropriate content to the user. Also, the stored statistics can be used to display the overall statistics for each user in the teacher and parent console.

6.2.1 Teacher console display

The statistics that should be displayed per class per user according to the requirement specification are:

- Current preferred word level with history
- Statistics per played session:
  - Session date
  - Session playtime
  - Occurred stages
  - Occurred target words

  - Occurred actors

- Statistics per occurred target word:
  - Target word in game occurrence count
  - Target word in game contextualization count
  - Target word completion, based on priority

However, the statistics screen is not strictly limited to these statistics. Basically any information regarding the played game can be retrieved, because all statistical data is stored in the database along with all the other game content such as the detailed information about each stage including the involved actors, possible solutions, available plugins and so on.

The data about the session that is displayed is only limited by the needs of the interaction designer. The interaction design and development of the teacher console was still in progress when my internship ended so I have no further information on the status of this component.

6.2.2 Statistics data requirements

To display these statistics in a teacher console, some data needs to be available that was gathered during gameplay. By creating statistics data structure at the start of each individual game session, statistical data can be gathered and stored in this structure throughout the session. When the session is done, the final scores are calculated and added to the data structure just before it is sent to the server for storage and processing.

This client-side generated statistical data structure will at least contain:

- Array of student ids that are participating in this session
- The total time played during the session
- Array of played stages, containing the following data for each stage
  - The stage id
  - The selected target word (plugin) id
  - Target word occurrence count
  - Target word contextualization count
  - Stage solution score

6.3 Gathering statistics client side

In this section, the creation process of the aforementioned stats data structure will be described in detail. The data structure is created on the start of the session and transmitted at the end of the session after all information is registered, calculated and added to the structure.

6.3.1 Creating the statistics package

- When starting a session, a new statistics structure is created
- After creation, the student array is filled with the participating student ids
- Each time a stage starts, a new stage structure is created within the session structure
- Each time a sentence is played by the game engine, the sentence id is recorded in the data structure
- Whenever a target word or contextualization sentence occurs, the appropriate counters in the data structure are raised
- When the stage is over, the solution score is calculated and added to the data structure

6.3.2 Gathering the data for the solution score calculation

1. Every time a stage starts, a solution score structure is created to maintain the data required for the solution score calculation.

2. Each time the user draws an object when a specific instruction is given, the *match score* difference with the right solution is stored. The match score is used in the current game to decide whether a drawn line matches the features of the line corresponding with the right solution. Also, the drawn objects counter is increased to calculate the average match score difference later.

3. Every time a contextualization sentence is played by the game engine the contextualization counter is increased in the solution score structure.

4. The auto solve function of the game is used to automatically solve conflicts when the student fails to do so within a certain time span. Whenever the auto solve is called, an auto solve flag is stored in the solution score structure.

5. After each stage the instruction index and loop count are stored. The instruction index is a value that corresponds with the amount of instructions that are given to the student before the right answer is drawn. At the lowest instruction index, Haas might question the player to draw a solution while at a higher index Haas might ask the player to draw a long line from the ground up into the air. The loop count is a value that records how many times all of the available instructions are given.

Using the gathered data, the solution score can be calculated at the end of each stage using the algorithm described in the following section. Please note that during testing, these algorithm variables will be fine-tuned to produce an optimal effect.

6.3.3 Calculating the solution score

The solution score is an indication of the ability of a student to perform the stage's problem adequately and in a timely fashion. This score is used by the ITS to see how well the student performed on a certain stage and influences the change that is made to the preferred difficulty.

The solution score is calculated in three separate parts using the aforementioned gathered data. The three separate parts add up to the final solution score. The solution score is always a value between 0 and 1.5 and is added to the statistics structure at the end of each stage. An overview of all mathematical equations used in solution score calculations can be found in appendix section B.1.1

1. The first and largest part of the score (*s1*, equation B.1.1.1) can be obtained by getting a low instruction index. For this part, a maximum score of 1 can be awarded, subtracted by the instruction penalty of 0.1 point subtracted for each instruction index above 3. Please note that when the auto solve is called, 0 points are rewarded for this part.

2. The next part of the score (*s2*, equation B.1.1.2) is awarded according to the average difference between the match score of the drawn object and the requested object. For this solution score part a maximum of 0.5 points can be awarded. The maximum amount of points is multiplied with a value between 0 and 1, calculated from the average match difference.

   To ensure that students are not heavily punished for drawing a very wrong object with a very high match difference, a maximum difference constant is defined. This helps developers regulate the punishment effect of this part of the algorithm. Please note that when right objects are drawn, the match score difference is always 0 to ensure that only count wrong objects are counted.

3. The last part of the solution score (*s3*, equation B.1.1.3) is calculated according to the amount of contextualization sentences that were played back during the stage. This part of the score can be seen as a bonus for the passive action of listening to contextualizing sentences, even when the student fails to complete the given tasks. For this part, a maximum score of 0.1 is awarded through $0.1/10 = 0.01$ point per contextualization sentence.

Each of the separate solution score parts is added up to the *solution score* (B.1.1.4). This result is stored in the statistics package for this specific stage and plugin combination.

6.4 Processing statistics package server side

After the statistics data structure is finalized, it is formed into a packet and sent off to the server back end through an AMFPHP service call. When the created statistics package arrives at the server, the server processes the package and sends an acknowledgement object back, containing a status message.

The information that has to be updated on the server side is linked to the specific students through student id's. The first field of the stats package contains an array of student id's that are associated with that session.

The following field contains an array of handled stages, each having a session id, the handled target word id, the amount of times the target word has occurred, the amount of times the target word was contextualized and the solution score associated with that stage. These data are then inserted into the database, linked through the student id.

The solution score matched to a target word is then used in conjunction with the occurrence count to determine the priority of that target word. This priority field is used in later sessions to determine which stages are on top of the list next the time a stage is loaded.

6.4.1 Dynamic Difficulty Adjustment

To adapt the difficulty of the game to the level of a student, we calculate a new preferred word level for that student using the submitted solution scores, averages from the available student data in the database and several weightings. In this subsection the procedure to calculate the word level adjustment is discussed. The mathematical equations corresponding to these steps can be found in appendix section B.1.2.

1. For each submitted stage;
   (a) We start the procedure by determining the *average solution score* (B.1.2.1) of *n* similar skilled students, that have recently played this stage and plugin. This gives us a stable value to compare the current students score to.
   (b) We then compare this average with the measured solution score and note the *solution score difference* (B.1.2.2) for this stage.
2. To determine how significant the solution scores are, we determine the *average mismatch* (B.1.2.3) for each stage by subtracting the chosen plugin difficulty from the current preferred word level of the student. By taking the average of these mismatches over all of the *n* played stages, we can determine how significant the average solution score differences are, related to the difficulty level of the student.
3. Eight different significancy cases are separately handled: score a lot better, a little better, a little worse or a lot worse than the average on a more difficult or less difficult stage. In these cases we use the average mismatch to adjust a modifier. The final word level adjustment is multiplied by this modifier. We can now alter the significancy of a word level change by setting the modifier value between zero and one according to the aforementioned cases.

4. The average of the solution score differences is calculated and used in an equation in which the averages of the previous four and the current solution score are averaged using a weighting system. In this weighting system the most recent average contribute more to the average than the older ones. This average, called the *weighted frame average* (B.1.2.4) is now used to determine whether the students preferred word level will decrease, increase or stay the same.

   An example of this calculation is demonstrated in figure 6.2. In this example, we see the frame of contributing values shifting when a new value is added. Each fame value is weighted separately, with increasing weight values towards the newest added score difference.

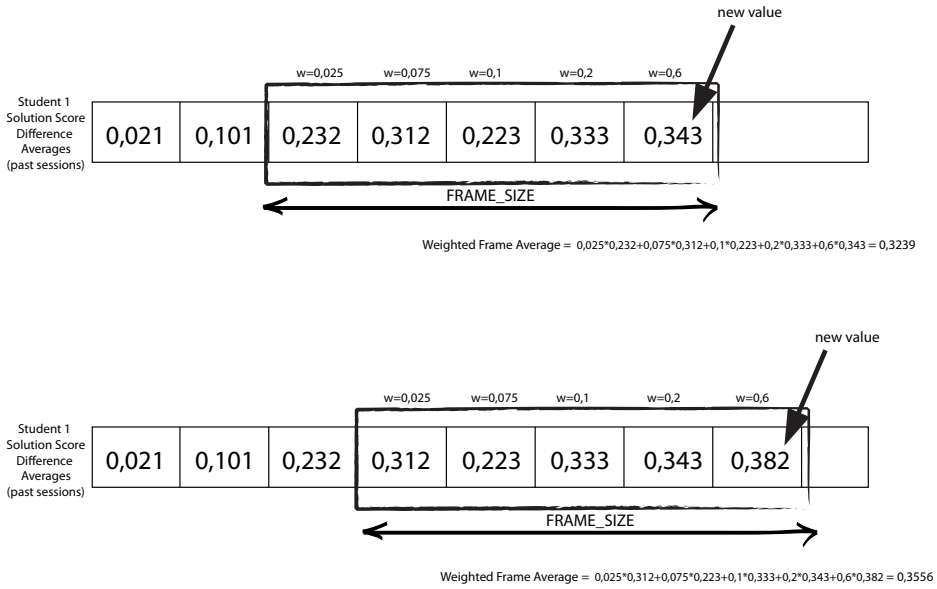

Figure 6.2: Frame shifting after new score is added, showing the weighted frame average calculations before and after the frame has shifted.

5. To decide upon the change of preferred the word level, we create a "comfort zone" between two weighted frame average values: the lower bound and the upper bound (B.1.2.5). When the current weighted frame average is between these values, the preferred word level stays the same.

6. When the frame average is below or above these values, the preferred word level is increased (easier content) or decreased (harder content) respectively.

7. The amount with which the word level is increased or decreased are called the increase rate and decrease rate, which have an effect on the speed at which student's preferred word levels are adjusted.

6.4.2 Target word priority calculation

To calculate the priority of a target word after receiving the statistics package from the client, both the target word occurrence and solution scores are evaluated in the following procedure. The mathematical equations discussed in this procedure can be found in appendix section B.1.3.

1. First off, all target words start with zero occurrences and a priority of 1. This makes all the priorities of target words that have not occurred yet equal at the start of the students progress history.

2. The first decision made to decide which target word will be used in the following stage is based on the the occurrence count of that target word. The occurrence count is then compared to the minimal and maximal occurrence constants to determine whether the minimum and maximum amount of occurrences are already reached. By manipulating these constants we can effectively control the amount of repetition for each target word.

3. When the minimum amount of occurrences isn't yet reached, we have to immediately increase the priority of this word to 2, to ensure this target word is on the top of the priority list the next time a stage is selected.

4. When the maximum amount of occurrences is reached, another equation is used (B.1.3.1). In this case, the priority is set to a certain value lower than 1 (below the initialization value) based on the amount of occurrences. The more the target word is repeated, the more the priority is decreased.

5. When the amount of occurrences lies between the minimum and maximum occurrence constants, we have to evaluate how well the student has done in order to determine whether the word should be marked completed or that it has to be repeated a bit more. This is done by comparing the awarded solution score with the average of the last n students, calculated in a similar manner as the described procedure in paragraph 6.4.1.

   This solution score difference value can now be used to evaluate how well the student has done compared to the average of the last n students. If the solution score difference value is below -0.2, the target word will have to be repeated some more, though it is less important than a target word with priority 2.

   Therefore, we have to subtract a small amount of the priority value decreasing it below priority 2 but above priority 1. To do so, we calculate a *priority delta* (B.1.3.2) which will be subtracted from the current priority based on the amount of occurrences while ensuring the priority value is between 1 and 2.

6.5 Stage and Plugin selection

When a "loadStage" remote call is made from the client to the server, the server has to respond by fetching content from the database tailored to the progress and level of the current student. Content can be fetched without any calculations, because when target words have occurred before, their priorities are preset and pre-calculated and the plugins are directly linked to word levels. We first sort the list on priority and then on on word level, when picking the content from the list of suitable, available stages and plugins.

6.5.1 Based on priority

Though we want the content to be tailored to the preferred word level of the student as good as possible, high prioritized words are always picked before others, for it is crucial that these words are repeated in the following sessions.

6.5.2 Based on word level

After the selection on priority, the next filtering step picks the plugin with the closest word level to the preferred word level of the current student.

6.5.3 Based on stage type

The final selection step is made based on available stage types. For each consecutive occurrence of a target word, a different order of stage types is used, fitted to provide the best possible context for a word on a given time.

The game allows three different stage types as of yet. The first stage type is the *neutraal* stage and is considered the easiest, for the player does not have to draw anything themselves in this stage. In this stage type all visible and audible context is given without interaction.

The second stage type, the so called *meer* stage is the stage with medium, for this type stage asks the student to draw more of the objects which are already visible on screen. Because there are already examples of the solution online, this stage provides clear visual context for the solution.

The last stage type, the so called *geen* stage is considered the hardest, because the student is asked to draw an object that isn't represented visually in the stage yet. This is much harder because the student has to think of the right solution, visualize it and draw a line on the screen that best resembles the size and location of that solution.

The preferred order of the stages based on stage type depends on the amount of occurrences of a certain target word. When the target word is new and unknown, easier stage types should be used, though the harder stage types can be preferred when the target word has occurred several times before. By altering these stage types, we can present the same stage with a slight alteration multiple times to the same student while keeping the problem challenging.

The order of stage types per amount of target word occurrences is defined as follows:

1. When a target word occurs for the first time, the preferred order of the stage types are first the *meer* stage, then the *geen* stage and finally the *neutraal* stage

2. When a target word occurs for the second time, the preferred order of the stage types is first preferably not the same stage as the first occurrence, then the *geen* stage, followed by the *meer* stage and finally the *neutraal* stage.

3. When a target word occurs for the third time or more, we preferably pick a different stage from first and second occurrence and if this is not possible, pick stage type at random.

6.6 Database Architecture

Rewriting and expanding the MniH database was one of the key components of the implementation of the Intelligent Tutor System. Therefore, much attention was given to the engineering of the new database during my internship. Though many data could be reused from the current system, the system has undergone a radical overhaul in both consistency and functionality.

Before using any language to implement the database, an entity relationship diagram was constructed that provided a clear overview of the data, combining the different essential parts of the system into one big relational database. As seen in the appendix figure C.1, entities and their properties are displayed in rectangles and relationship tables in diamonds. All tables always contain a primary key called ID to identify the values in the database and underlined properties are foreign keys that link to another table.

The database is split up in several shaded areas that describe different parts of the system. Each area is used by different parts of the system, though some parts of the system use data from multiple areas. These areas consist of game logic, statistics, contact info, educational, world members and script actions. The following sections will give a brief overview of these database areas.

6.6.1 Game logic

The game logic area consists of data that is directly used by the game engine as content for the game. This part of the database contains all data that was previously present in the pre-ITS version of the game. Within this area several key components of the game such as the gameplay situations (stages) with their solutions and possible environments (soil / relief / climate) are stored.

In addition, the new separated system to store dialogs and script actions separately is implemented in this area. In the previous version, script actions such as animations and camera actions (called ANI's internally) were hacked into the database disguised as a special dialog that are recognized and handled separately in the game engine. The improvement offers a clear difference between script actions and dialogs and is more flexible to extend.

6.6.2 Statistics

The entities and relations in the statistics area are used to store data gathered during gameplay sessions for specific students. These statistics are used to determine the desired difficulty for the students and to provide the ability for the teachers front end to give insight into the learning progress of the student. In each session, several stages are played that all have one specific plugin (target word) selected that is matched to the preferred difficulty level of the student. For each student, all occurred words are registered separately along with counters that provide some statistics that can be displayed as progress.

6.6.3 Contact info

The contact info area contains information about schools, classes, teachers and students. This part of the database wasn't developed further, because an online test version of MniH is currently being tested in which the same type of information is stored. When the online test version succeeds, that database will be merged with this area to keep track of the details about the systems users and affiliations.

6.6.4 Educational

This small but important area of the system holds the educational domain data that is meant to be taught. This consists of carefully crafted word lists of target words divided in several themes by educational experts. As a complement to this area, the sentences used in the dialogs provide contextualizing sentences for these target words.

6.6.5 World members

The world members are the database equivalents of objects and actors that appear on screen in the game. Due to the new script actions being generic, we have devised the "world member" entity that can be any one of the mentioned classes such as actors, flat characters, objects and abstract targets. For example, a zooming script action can be targeted at an actor, but also at an object or a group of objects.

6.6.6 Script actions

The script actions area of the database diagram is created to facilitate the back end script and scenario tool. Though script actions are instantiated in the "actions", the possible script actions have to be defined per group, including the possible actors and targets per action. These entities allow drop boxes containing the predefined possibilities per action in the script and scenario tool.

CHAPTER 7

System implementation
Technology, system overview and testing

When the systems architectural design was finished, the implementation of the MniH ITS started. Using all previous documents, models and diagrams, the developer team was able to start implementing the system in the most suitable technologies. I have spent my remaining internship time on the implementation of the server side logic, because my project focused most on the ITS server logic.

This implementation chapter consists of an overview of the used technologies and design patterns, a detailed explanation of the code flow diagrams that were created during the implementation process, a description of the testing procedures that were used to debug and check the workings of the system. Finally, I will give a short overview of the implementation sidetracks I have taken during my internship to support content writing decisions by offering tools to produce database related statistics.

7.1 Technology overview

The following overview of the technologies that were used for the implementation is written from an information scientist point of view, who is focused on the goal of the system: "What technologies do I need to adequately realize this system?". To answer this question, I will describe the arguments for every technology that was used to realize the different components of the ITS while keeping in mind the scope, the time schedule and the available budget.

7.1.1 Actionscript 3 for client side

The first choice for Actionscript client side code was the most obvious decision to make because the current game structure is already written in Actionscript and there are just a couple of minor adjustments that have to be made to the code base to enable statistics gathering and sending.

The game client can also be ran in a web browser window, because Actionscript code compiles to Adobe Flash runtime code, executable by a browser flash plugin. This eliminates the need of physical installation discs, dvd-rom readers and installation issues due to rights restrictions on school PCs. Another big advantage of Flash is that the software used to run the code is available on multiple platforms and doesn't require any porting.

Unfortunately, there are some limitations to the Actionscript language and Flash, the most important one being the performance capabilities of this technology. To enable the code to be executable on multiple platforms, all code is executed through the Flash Player's Virtual Machine which is yet another interpretation layer between the software code and the hardware it is executed from.

The biggest disadvantage of the Flash platform is the Flash compiler that is very inefficient [14]. To ensure backwards compatibility with the oldest versions of Flash, old-fashioned routines are still implemented and used. A lot of performance could be gained with optimization for hardware acceleration, which has been introduced in Flash Player version 10.1 in a very limited fashion, supporting only h264 video playback and the use of GPU on supported mobile devices, which unfortunately doesn't increase performance across all platforms.

7.1.2 PHP back end web service

Using PHP to implement the ITS API and back end services allows MniH to quickly setup a scalable, fast, stable and secure web service without necessitating any expensive licenses and costly licensed developers. The current developer team has been using PHP before in the first online test versions of MniH which has proven to be reliable and easy to use.

Creating our back end system in PHP also has the advantage that it is easy to create a connection from Flash applications through the AMFPHP communication framework, which will be discussed in the next section. The required ITS web service is very heavily integrated with database access for statistics storage, retrieval and processing. This is another reason why we have chosen for PHP, because it offers an excellent and fast interface for databases such as MySQL, which will be discussed in a later section.

7.1.3 AMFPHP client - server communication

AMFPHP is a PHP framework that allows for easy RPC connections to PHP frameworks from Flash applications. By using serialized data and a compact messaging notation, data streams are lightweight and fast, which was the primary requirement of the communication technology.

We are able to construct Flash objects server side and send these back to the client, ready to be used in the code without requiring additional type casting or parsing because of AMFPHP's specific implementation of Flash remoting. In this way, the path of database information to the client is as short, lightweight and structured as possible, allowing us to serve many connections with the shortest possible processing time.

7.1.4 MySQL database software

We have chosen for MySQL as server side database technology. It works perfectly in conjunction with PHP using the fast MySQLi interface module of the web server running PHP. Furthermore, MySQL is well known for of its speed, easy replication, ease of use and the vast amount of documentation and code examples. Because MySQL is open source, it doesn't require any licensing costs.

However, during the development we have found ourselves in need of some advanced database functionality that wasn't available in MySQL, which resulted in a little less reliable software and some workarounds that wouldn't have been necessary in more advanced database systems such as MSSQL or Oracle. To get some advanced functionality such as transactions and rollbacks, we could have upgraded the MySQL database structure to the InnoDB format. However, the hosting provider did not support this technology.

7.1.5 PureMVC model view controller pattern

To keep the web service back end consistent with the Actionscript game code, we have chosen for the PureMVC design pattern. PureMVC is lightweight framework for creating applications based upon the classic *Model-View-Controller* design meta-pattern which creates a hard separation between application logic, input and representation and data representation.

This pattern is used to create a clear distinction between code purposes involving either the system's data (model), user input and output (view) and the logic that operates on both (controller). Another big advantage of using this model is that certain parts of code, such as database access, can easily be reused in different parts of the system. For instance, the same database connector code is used in the ITS storage component and the retrieval of data for the online content authoring tool. This also allows easy code maintenance such as updating the MySQL connector to a newer version without affecting the original code interface.

7.2 Back end structure PureMVC Architecture

The created back end structure is considered mostly with model and controller based code, because there is no real graphical user interface (GUI). The only view that is used in the system is the so-called "output mediator" which sends the responses to the client side calls through http back to the client. The following sections show several PureMVC diagrams that represent the implemented PHP code structures.

7.2.1 MatchStage

The first part of the back end implementation is the *MatchStage* call that is made by the game client to retrieve the next stage and plugin combination that is most suitable for the current situation. The retrieved stage contains the dialog and script actions as well as the solution that has to be drawn.

When creating this module, we have decided to merge some of the existing calls to retrieve data. The original game communicated with the (local) database several times to first check the situation, then match the stage and then load the content separately. To improve the throughput of the system and decrease the communication overhead, we have decided to merge these calls into one *MatchStage* call that does all these things at once.

An important note has to be made here. The existing game has to be adapted to the new way of communicating between server and client, because the dialog and actions are now separated. Furthermore, a script actions queue will have to be devised to parse and queue the incoming script actions, but this was not really relevant in my part of the project.

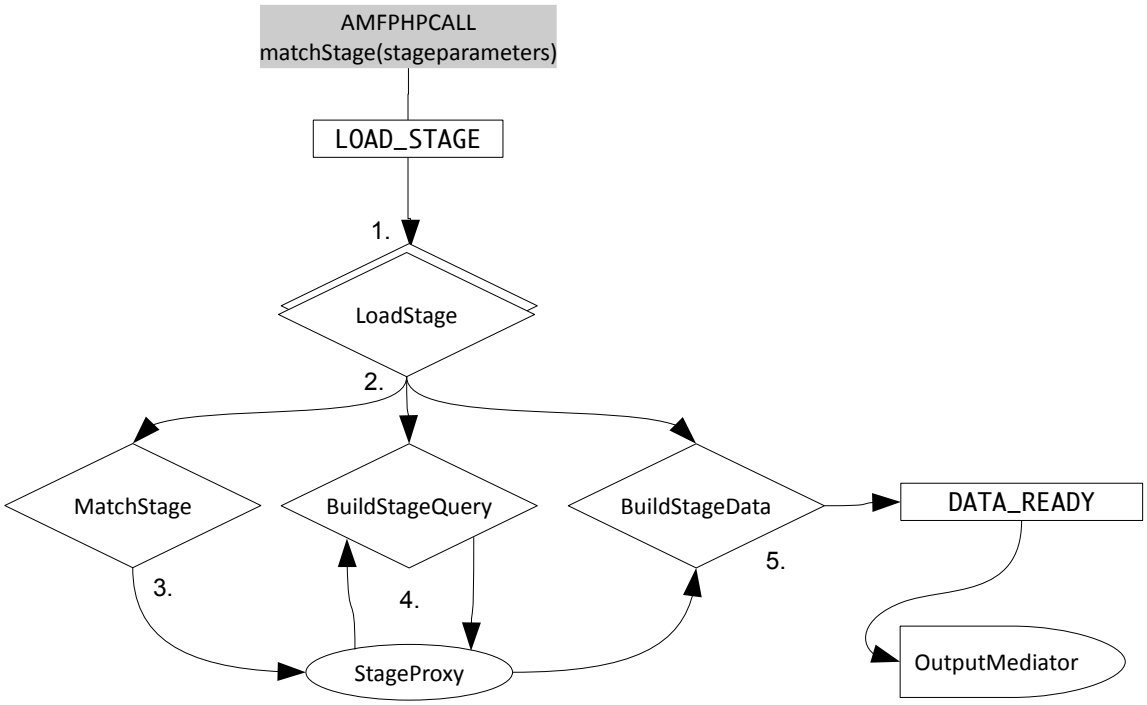

Figure 7.1: ITS stage matching backend service PureMVC PHP architecture.

Figure 7.1 gives a clear overview of the execution flow when this component is used.

1. The client side Flash application calls the PHP PureMVC structure through an AMFPHP request to the "loadStage" service. This triggers the PureMVC PHP application to load the *LoadStage* macro command.

2. *LoadStage* adds three subqueries that sequentially follow each other: first matching the

stage in the MatchStage command, then building the stage retrieval query and finally building the retrieved data to an object that is returned to the client.

3. *MatchStage* stores the matched stage and plugin id to the StageProxy
4. *BuildStageQuery* retrieves the matched stage/plugin id from the StageProxy and fills the stage query according to these ids and the initially sent parameters.
5. Finally, in the *BuildStageData* command the built query is executed and the data is built into an AMF response that is returned to the client through the OutputMediator.

7.2.2 Sendstats

The second part of the back end implementation is the *SendStats* call that is made by the game client to store and process the incoming game session statistics. When the processing is done, the word level of the attending students is changed and an acknowledgement response is sent back to the client.

The main focus of this component is to efficiently and quickly store and process the given statistics into session specific statistics. This is done by logging the session in the database, raising target word related counters and raising the total play time in the database for each student. Finally, the dynamic difficulty adjustment algorithm mentioned in section 6.4.1 is executed to adjust the word level of the students to update the learner model of the system.

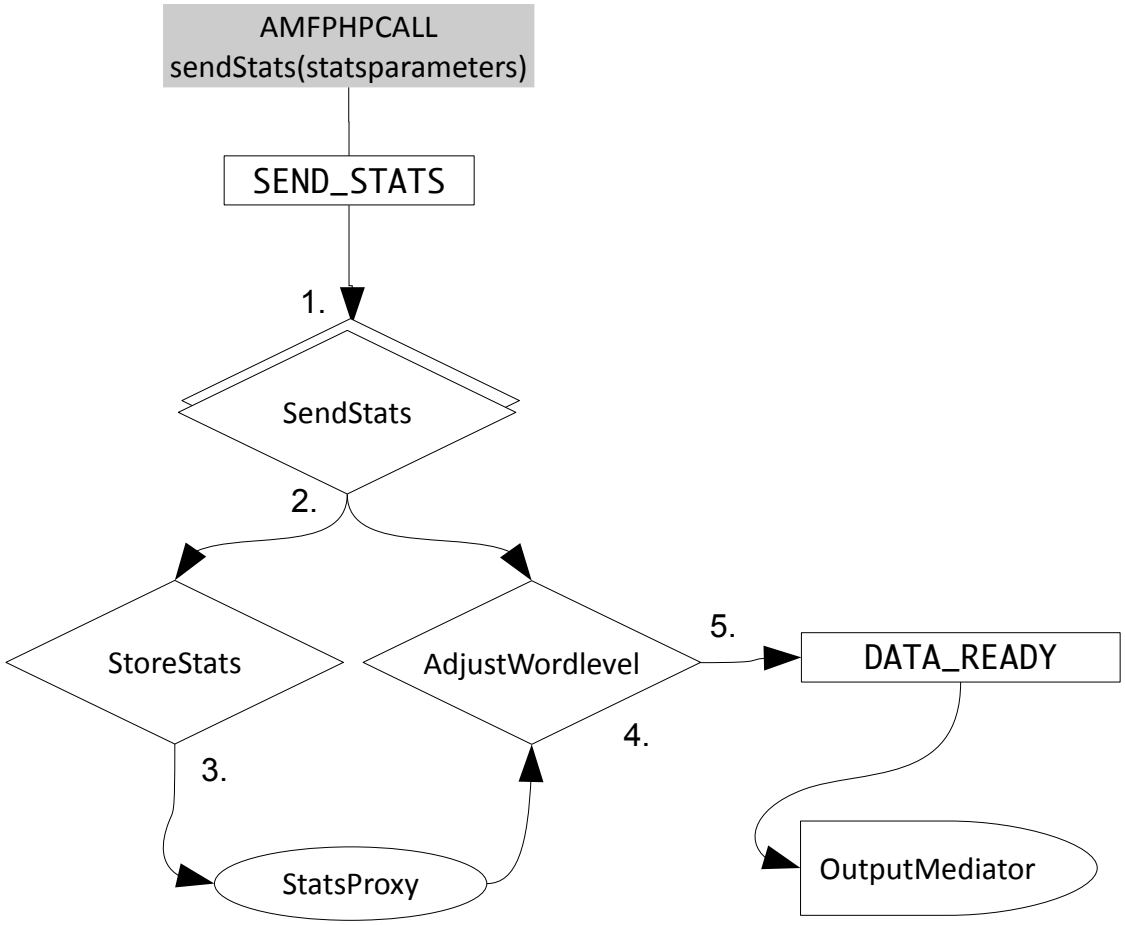

Figure 7.2: ITS stats sending backend service PureMVC PHP architecture.

Figure 7.2 gives a clear overview of the execution flow when this component is used.

1. The clientside Flash application calls the PHP PureMVC structure through an AMFPHP request to the sendStats service. This triggers the PureMVC PHP application to load the *SendStats* macro command.

2. *SendStats* adds two subqueries that sequentially follow each other: first the transmitted statistics parameters are processed and stored in the database and then the new word levels for the concerned students are calculated.

3. *StoreStats* processes and stores the transmitted statistics in the database and saves the session id in the stats proxy for the next command

4. *AdjustWordlevel* calculates the new word levels for the students concerned according to the algorithms defined in the architecture document.

5. Finally, the response acknowledgement data is built into an AMF response that is returned to the client through the OutputMediator.

7.3 Testing / Debugging

During development, debugging and testing is essential to a successful implementation. It was essential to find a tool that could help me find out what exactly is being transferred, because the main functionality of the web site back end is the communication between the server and the client.

As a test environment, I have setup a local web server running the development version of the code with a local database which allowed me to test communication to a server on my machine. To test the interaction with the game client, we had to find a replacement for the actual game, because the implementation of the client side stats gathering and sending was scheduled to be developed after my internship ended.

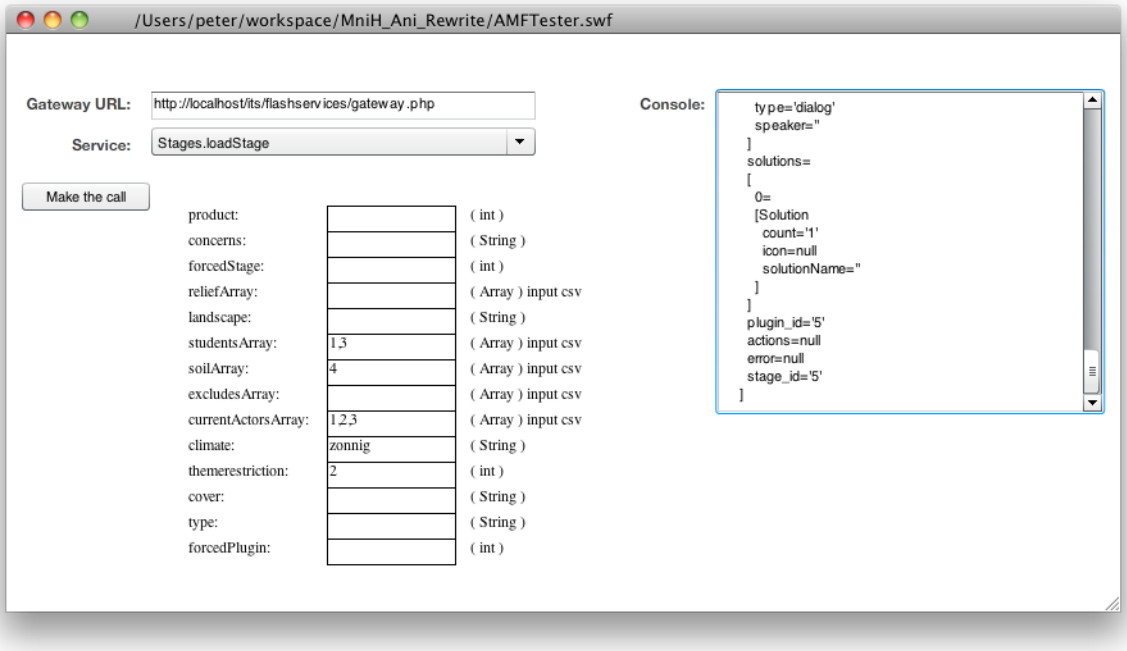

Figure 7.3: AMFtester Flash application that mimics client calls to the server back end for development, debugging and testing purposes.

As a placeholder for the ITS enabled game client, the MniH team has developed the "AMF tester jig" which is a Flash application that mimics all of the possible requests that can be sent out by the game client. As seen in figure 7.3, developers are able to specify a gateway url, to

select the desired service that has to be debugged and to fill in the parameter package that is associated with the selected call. After the parameters are filled in, the developer clicks the "Make a call" button and sees the web service response in the console window.

Though this is a great tool to manually craft request packets that are sent to the server, it is very useful to see what actual data is transmitted over the wire. For this exact purpose, I have used the Charles web Debugging Proxy[1]. Charles is an HTTP monitor that enables developers to view all of the HTTP traffic between their machine and the Internet.

A very helpful feature of Charles is the ability to repeat previously sent packets, which is extremely useful when debugging a specific service feature. It is also possible to store the packet, originally from the AMF tester, for later use. This avoids having to manually fill in the parameters each time when developing one certain service feature.

To debug the statistics call, the AMF tester jig did not suffice, because the *SendStats* call is made using run time gathered data in a complex data structure. To allow the debugging of the statistics sending I have extended the AMF tester with functionality to semi-randomly generate stats within a certain range and sending these to the server using the '=' keystroke. This was also very helpful when stress testing, because random statistics packets can be send in rapid succession, emulating a lot of students using the service.

7.4 Implementation sidetracks: Content tools

As a side track I have developed several mashups during the design phase of the project to support the content developers with their decisions about the ITS based on the current database information. These tools, internally called "content-tools", are PHP / jQuery mashups that display statistical data about the current content database. From the several tools that have been developed, I will briefly discuss the two most used tools in the following sections.

7.4.1 Wordcount tool

The first tool, seen in figure 7.4 is the word count tool that lists all the words present in the dialogs of the game, counts the words and shows the difficulty of that word according to the wordlist by Schaerlaekens [18]. When a word is clicked, in this case the word "teken" an overview of all conflict steps is given, showing word instances per conflict step. The pie and bar charts are generated by Google via the Google charts API.

With this tool, users can get some insight into the statistical data on the currently available content. First, the user is provided with a better overview of the most and least used words. Second, the user is able to see the difficulty of most and least used words, which helps to form a statement about the general difficulty of the game. Third, user can quickly search for a word in the search box to quickly look up the word count and difficulty of that word.

7.4.2 Difficulty parser tool

Another content tool seen in figure 7.5, is a web application quite like the previous tool and displays the same information, but only for the words that are filled into the input field. This allows users to quickly get an overview of the difficulty of a sentence based on the separate word difficulty. Though indexing the difficulty of a sentence can and should be done using a big array of tests, carefully described in an excellent paper about a difficulty indexing tool by Kraf and Pander Maat [16], but using these techniques to determine the difficulty of a sentence was far beyond the scope of the content tool.

[1]For more information about Charles visit the website `http://www.charlesproxy.com/`

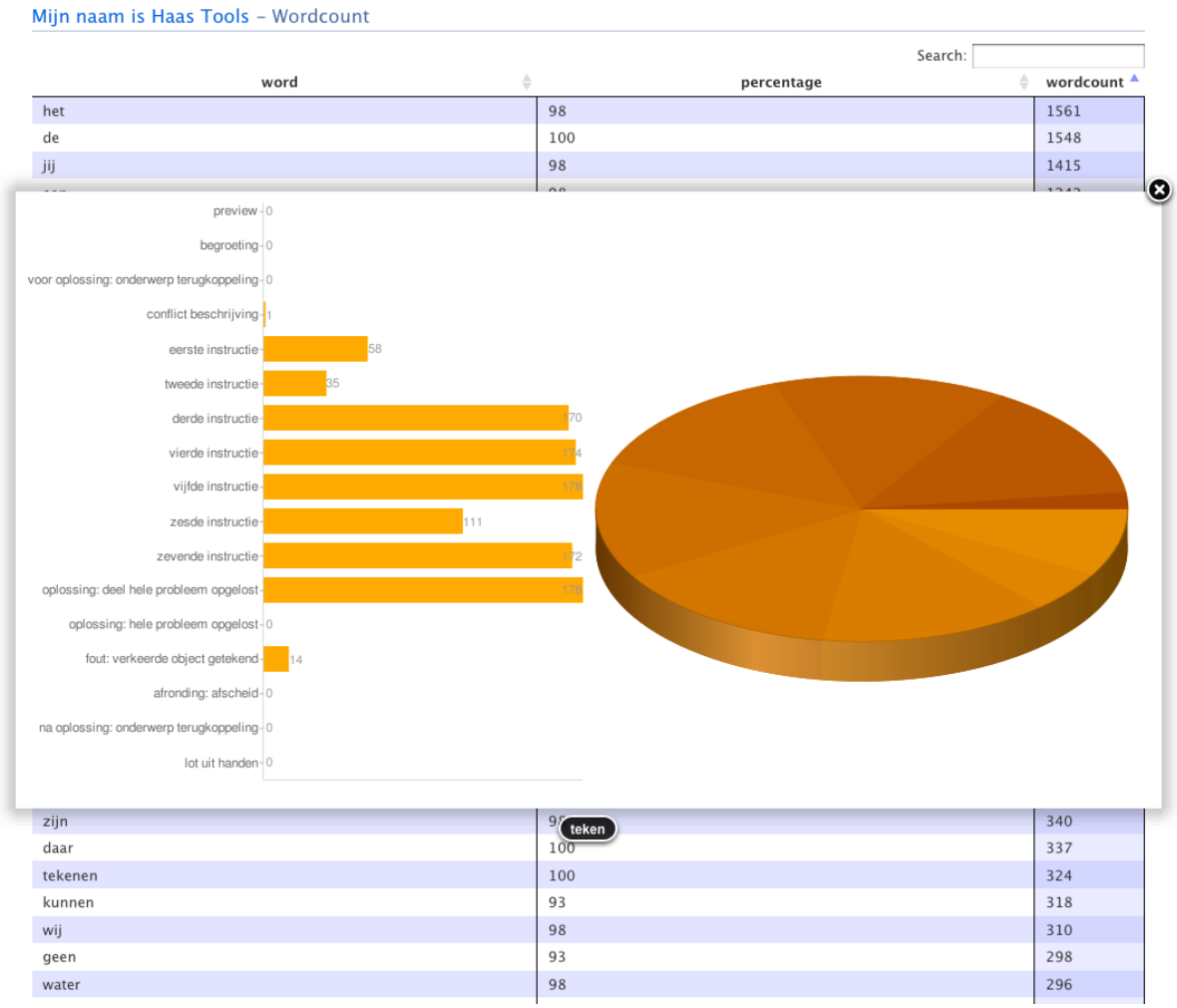

Figure 7.4: Wordcount content tool that counts the occurrences of a word for each separate conflictstep and shows the difficulty per word.

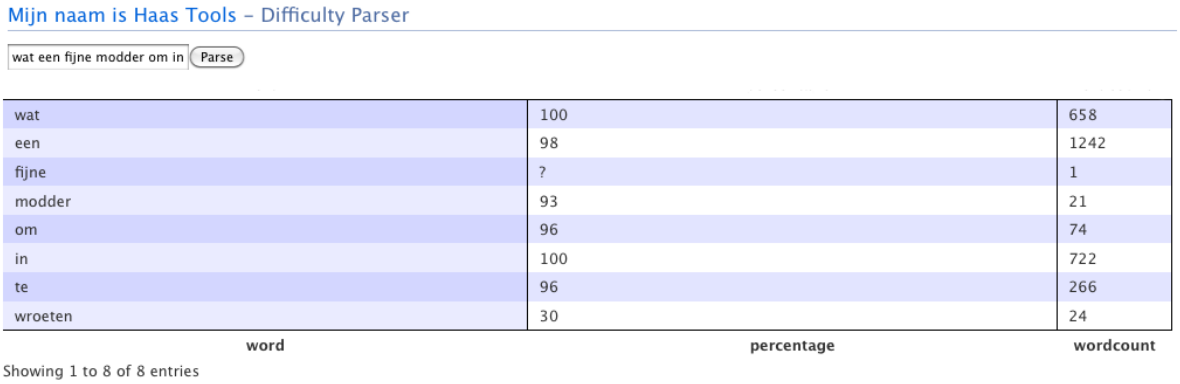

Figure 7.5: Difficulty parser content tool that parses a sentence and displays the difficulty percentage and word count of its individual words in the content database.

CHAPTER 8

Evaluation

Theoretical and computational models to implementation

To properly finish the internship and thesis, it is important to evaluate the methods used to accomplish the project's tasks and goals. In this chapter, an evaluation is given of the discussed topics such as the creation of the conceptual models, forging these models into an architecture and the resulting implementation process.

Furthermore, some remarks will be made on the generalizability of the solution, the pitfalls that I encountered during the project and the future development possibilities for this project. The final part of the chapter consists of a reflection of the internship supervisor and a personal reflection on the whole internship.

8.1 Realization of design vision to conceptual models

The majority of the conceptual models were created from the vision documents and the brainstorm discussions with the developers. This has proven to be an excellent method, because all participants are pro-actively thinking about the problems and solutions to address a certain functionality.

While determining the goal of the system through the vision document and the instructions given by the supervisors, it was very pleasant that there was a lot of room for my own ideas and thoughts about intelligent tutor systems in general. The supervisors, developers and the EN were very open to new approaches which resulted in a very healthy and productive atmosphere where everyone could express their wants and needs.

Because the background research was done very early in the process, I have quickly gained a lot of knowledge about intelligent tutor systems. Combined with my experience in software engineering and software development, my expertise was a great support for the development team during the brainstorms and the development process in general.

In general, the feedback of the EN during the educational justification meetings was very helpful in the refinement process of the conceptual models. Only a certain number of conclusions can be drawn on the educational effect of certain methods, because educational methods in general are still quite a new area. This offers a lot of possibilities for experiments on new and innovative ways of tutoring such as the tutoring system implemented for the MniH serious game.

8.2 Realization of conceptual to computational and architectural model

The process of creating computational models from the conceptual models was a gentle process of proposition, feedback and refinement. Once again, I have had a lot of freedom while devising certain models, which had a great effect on the creative process of creating a sufficient and justifiable system within the given time span.

Devising the architecture of the system was a very interesting challenge of finding a way to achieve all functionality while remaining a structured overview of implementable components. Though the general idea of the architecture was formed quickly, the details were

refined over and over again during the process of presenting, acquiring all sorts of feedback from different involved people.

Unfortunately, a lot small details had to be considered during the development process, including many details that didn't have much to do with the ITS logic per se, because the architecture of the new ITS enabled game structure radically differs from the original structure. This backwards compatibility and reorganization of original structures took much time to process, which, from time to time, was quite a distraction from the main project goals.

8.3 Implementation of the architecture

The implementation of the system was well prepared with the detailed architecture, carefully describing the structure of the components and their individual inner workings. The close cooperation and communication with the other developers has enabled us to divide the work and create a schedule to implement the ITS in time for the deadline. By separating the system into multiple well engineered functional chunks of the system, individual tasks were easy to assign.

The database model was a great task to work on, because it allowed me to create a structured overview of everything that has to do with the game. By abstracting each possible game related entity, I quickly learned every little detail about the current system and the future system. It was also a great experience to apply the modeling techniques I have learned in classes to transform the existing, mashed up database into a consistent and structured whole using proven engineering methods and modeling techniques.

Improving the communications model of the game and the database, which by now has evolved into a full fledged client-server model was a tricky trade-off between overhead and latency. By combining certain calls into one, less communication overhead was necessary, though the server side calls would get heavier and required more communication with the database.

8.4 Validation

In this section, two big parts of the system's validation will be discussed. The first subsection will be about the intelligence tests that were discussed in chapter 3, and explaining why the MniH ITS satisfies these tests. The second subsection is about the testing procedure that will be performed at several test schools after all of the ITS is implemented which provides the developer team with usable statistics and offers a real life check of the educational effect.

8.4.1 Tests of intelligence

As promised in chapter 3, I will now evaluate the three tests of intelligence defined by Angelides and Paul, to see whether or not the MniH ITS can be classified as intelligent. The first test is whether or not the system has enough knowledge of the subject matter to solve problems in the domain of application. In our case, the ITS has a lot of carefully selected and structured knowledge on the material that has to be taught, and knows the correct answers to the problem the students have to solve. By observing the students while they play, the system can decide whether or not the right answers are given and how well the student has performed.

This leads to the second test of intelligence, which implies that the system should be able to deduce a user approximation of the domain knowledge. By observing and gathering statistics about the actions the student takes, we are able to make an approximation of the skills of the students. This is by far the weakest point of the system, because it is very hard to draw solid conclusions on the limited set interactions such as of drawing lines on the screen and responding to verbal instructions. Though the conclusions may not always be

as valid, we have strived to create an optimal educational environment, strictly following proven tutoring techniques, making the system as intelligent on this test as possible.

The final test is whether the system has a tutorial strategy that reduces the skill difference between the expert and the student's performance. By matching target words most suitable for the user and carefully semantizing and contextualizing these words in the games general dialog and instructional content, the student is able to get a better understanding of the words through visible and audible tutoring.

8.4.2 Test schools

Next to the test of Angelides and Paul, there will be a thorough effectiveness test of the ITS enabled MniH game performed at several test schools. In this test, half of the contesting schools will follow an ordinary curriculum while the other half uses a curriculum with 2 gameplay sessions of MniH per week per student. After these periods, a vocabulary test will be taken and the results of the two test groups will be compared.

During this test period, many student data will be generated and stored, which provides an excellent testing scenario for both stress tests and system effectiveness tests. During and after this testing period, developers are able to fine-tune all of the variable algorithms to produce the desired results. Developers will be able to see what kind of skills to expect from students and can change the algorithms accordingly, because a lot of algorithms use the student's average score.

Another very useful source of information are the teachers that use the system and explain the system to the students. By eliciting their desires and findings during the process, the MniH team is able to adjust the system according to their needs, producing an even more usable, friendly and clear system.

After these tests, the EN will process the results and produce an official review of the educational effect of the ITS enabled MniH game. Using this review, MniH is able to further develop the system and decide on what features have to be focused to make the system perform even better.

8.5 Pitfalls and Issues

Though most of the planning during the internship worked out fine, there were of course some pitfalls and issues that I encountered in the process. In this section, I will give a brief description of two of the largest issues that have occurred during the project, being the hard to scope size of the project and the new area of educational effect of the ITS.

8.5.1 Project scoping and time

First of all, the development of the ITS is a huge project, scoped into a relatively short period of time. Though there was still some time left to develop the ITS after my internship was finished, it still had to be done in a short amount of time, effectively limiting the complexity of the system to keep it implementable within the time span.

However, on several occasions during the design and implementations, even within the shortened scope there has been some serious feature skipping, because it would take too much time and would delay the strict schedule. Due to this extra scoping, the complexity of the system got decreased caused by the extra difficulties that were witnessed during the development itself.

8.5.2 New area of educational effect

Another major issue that is inherent to the development of an ITS is the new and uncertain area of educational content in games that are primarily focused on entertaining and immers-

ing the student. Even when applying known and proven methods in education, no guarantees can be given about the educational effect, on what form of education whatsoever. This has proven to be a great challenge during the development of the ITS and has constantly forced us to make important decisions on what we presumed would be the effect of the system.

Most of the time, especially when defining the values of variables that were used in the algorithmic models, we had to make an estimation of what we thought would be the correct value for a specific situation. However, these values might only just be significant for the average of this group, because we are talking about an extremely varied group of differently skilled children in elementary schools.

8.6 Future development

8.6.1 Current version

During my internship I have been able to develop most of the server side logic including the accompanying score / level calculation algorithms that were devised during the design and architecture phase. Though I have implemented a big part of this ITS architecture during my internship, there is still some development work that has to be done in order to get a complete and working ITS enabled MniH game.

First of all, all of the client side code in the game still has to be written, according to the guidelines described in this thesis. This includes the rewriting of the dialog and script actions queue, because of the new method of retrieving content via the online web service and the gathering, pre-processing and sending of the statistical data to the web service.

For the server side, most work has to be done in the script and scenario tool, which is the front end for content writers to insert ITS enabled content into the game's database. Even though this was not part of the scope of my project, this is an essential component for the ITS functionality and maintenance and also has a big impact on the effectiveness and educational value of the system.

Furthermore, the system as it is now should be thoroughly tested, validated and fine-tuned to match the need of the target audience. It is possible to guess certain values in advance to a certain degree, but it is evident that the system has to be tested in a real life environment by actual students in order to really be able to define the algorithmic values properly.

8.6.2 Future versions

Future versions of the ITS could possibly be far more intelligent and complex. During the initial design phase of the project we have discussed an abundance of features that would have a great impact on the interactional and immersive experience of the game and could greatly improve the educational value.

To scope the projects development work and create a reasonable schedule within the available timeframe, these functionalities have not been considered viable for this project. These could however prove to be very interesting to develop in the future. The system should be able to react upon these statistics far more intensively, because a lot of student statistics are gathered and processed into the learner model during the game sessions.

For instance, more complex models for repeating target words can be integrated in several parts of the game which would provide students with even more and subtle contact with new and unknown target words. Another worthwhile feature that could be expanded even further is the advantage of composing the content even more intelligently based on proven methods in educational text composition.

Even though there are a lot of possible improvements for the ITS, it has always been very important to keep in mind that the game should be fun to play. With every new feature of

the ITS that affects the gameplay, we have to consider the trade-off between student usability and the educational value of the feature. To create a truly immersive game, we should always strive to make the ITS as transparent as possible.

8.7 Future trends

In addition to the implementation of the ITS in the current game structure of MniH, there are a lot of future trends that could be of significant value to both the MniH brand and the educational effect of the existing products.

One of the upcoming technologies that might prove to be interesting for the MniH brand is multitouch for personal devices. Nowadays, people are expecting applications that run on multitouch capable devices to support multitouch input, because of the rising penetration of multitouch devices in society.

In this area, MniH has a lot of potential to support these multitouch movements by allowing multiple cursors to draw multiple shapes at once or more complex shapes using multiple fingers. This provides a fine opportunity to create extra challenging situations for skilled students, or to allow multiple students to use the software at the same time. By allowing multiple users to interact with the software at the same time, cooperation and communication are encouraged which in turn improves the educational effect of the program.

Another important trend in modern computing is the rise of smartphones that are capable of running complex applications with advanced user interactions and internet connectivity. The existence of virtual marketplaces that for devices like the iPhone, iPad, Windows Mobile and Android devices where device owners can purchase applications provide an excellent opportunity to sell applications worldwide, reach a large audience.

By creating a spin-off game or even just an interactive storybook with the MniH brand for mobile phones and tablets, the company can easily broaden its target audience and provide quick and easy access to the content. Mobile applications are usually smaller versions of a certain application adjusted for smaller screens, but can be enhanced using mobile phone sensors such as a gyroscope, compass, GPS location and internet connectivity.

With MniH future plans of internationalization of their brand, these marketplaces have a big potential to reach users all over the world. By providing English translations of the content in an attractive interactive mobile application, MniH can easily distribute their content all over the globe and extend the usage of their educational tools to the home market.

8.8 Reflections

The last piece of this chapter consists of two reflectional texts, one written by the MniH internship supervisor who is also part of the development and engineering team and one written by myself.

8.8.1 Mijn naam is Haas reflection on internship

Berend's reflection

8.8.2 Personal reflection on internship

In my five months of internship at MniH I had a wonderful experience working with very talented and young colleagues that all share a passion for innovative serious games and truly distinguish themselves from the rest of the educational industry. By creating a game in which children love to immerse themselves, MniH has found an innovative new media to interweave vocabulary expansion tutoring with a fun game to play.

By working closely with my fellow colleagues on a responsible and important task I have been enabled to help design, conceptualize, practice architecture and implement the ITS using my experience and knowledge from an ICT perspective. Next to the acquired business experience I have learned a lot about serious games, vocabulary expansion techniques and game code structure in general, truly making this internship a worthwhile effort.

I think the development process of the MniH ITS during my internship is a very valuable effort and can prove to be an effective tool in the improvement of the game's educational value. Overall I'm very satisfied with the goals that I have been able to accomplish during this internship, both implementation-wise, communication-wise and education-wise. I will most certainly not forget this wonderful experience and keep a close track on this innovative company.

CHAPTER 9

Summary and conclusion

The final chapter of this thesis consists of a short summary of the thesis content and an in depth conclusion of the research questions and thesis goals that were achieved. These sections can be read to obtain a quick overview of the whole internship and the results that were achieved.

9.1 Summary

The introductory chapter of this thesis gave a rough sketch of what the context of the research is and what the research questions of this thesis are. By giving a detailed description of the thesis structures, the readers have been able to get an overview of the several phases of the project. In addition, the reading guide provides an overview of the different target audience groups and gives these groups direct pointers to the right chapters of interest.

To offer a clear context for the ITS that would be engineered and implemented, the second chapter depicts the gameplay mechanics of the MniH serious game. By giving a step by step guide of the game, the reader is able to understand what the goal of the game is, why the game could be appealing to young children and how it is possible to enhance the game's educational value by implementing an ITS.

The following chapter about background research was written to scientifically substantiate the design and inner workings of the ITS. In this chapter I have investigated the properties of general ITSs, the possibilities of creating an abstract model of such a system and some information about ITS classifications and categories. Furthermore, this chapter contains some research to knowledge authoring, difficulty adjustment in games and an elaboration of the appropriate vocabulary expansion techniques.

The first internship phase consisted of establishing a vision, deciding upon the research methods and scoping the project. This phase is described in chapter 4, giving a detailed portrayal of each of the involved company teams and their involvement in the ITS development.

One of the most important parts of the project is depicted in chapter 5, showing the conceptual models for the internal ITS workings that were devised during the first phases of the internship. In this chapter we find elaborations on the progression techniques, how the difficulty will be adjusted, the scores will be gathered and how the tutoring model will be implemented in the current games dialog structure.

After the conceptual phase, the models were converted into an architecture in chapter 6. In this chapter, I have discussed several methods and diagrams that show the overall structure of the individual ITS components in the new ITS enabled game. Separated in a communications overview, statistics gathering, statistics processing, content selection and database architecture, these sections provide individual chunks that can be programmed in the implementation phase.

The final part of the internship was spent on the implementation of the aforementioned architecture. Chapter 7 is devoted to the justification of the technologies used to implement the ITS and shows several diagrams that give a more detailed view of the implemented code structure. The following part of this chapter explains the procedures used to test and debug the written code using the specially written test applications and Charles debugger. The final part of this chapter gives some insight into the implementation of the MniH content tools that were created to support decisions for content writers using statistical information from the content database.

The last chapter before the summary and conclusion is chapter 8 about the evaluation of the internship project. By discussing the realizations of the design, the conceptual models, the architecture and the implementation I have given some insight into the effectiveness of the chosen methods.

The following sections describe the validation techniques and the scheduled testing process of the ITS enabled MniH game that will be conducted at test schools, followed by a section on the pitfalls and issues that were encountered during this internship.

At the end of this chapter, we find an elaboration on future work, divided in future development of the MniH brand as well as future trends that could be explored by MniH to broaden their market and improve the educational value of their brand.

The evaluation chapter is concluded by two personal reflections, one written by the internship supervisor about my work at the company and one written by myself about my experiences and achievements during the internship.

9.2 Conclusion

With this thesis, I have created a scientifically substantiated report of the engineering process and implementation of the MniH ITS during my internship. Using this thesis, MniH can continue the work on the ITS and interested readers are brought up to speed on a real life example of an implemented ITS. Game developers might even be inspired to consider developing an ITS for their own serious game.

9.2.1 Goals

In this thesis many topics were discussed that are related to ITS development, including the background research of intelligent tutoring, design, conceptualizing, practicing architectecture and implementing ITSs. The goal of the ITS was to offer insight into the educational progress of the students and to adjust the content to the player based on theme and student skill.

These goals are well represented in the final architecture and implementation and the system has proven to be even more advanced than the original goal described, including advanced dynamic difficulty adjustment based on average skills and comprehensive game session statistics tracking.

9.2.2 Contextual research answers

This first contextual question posed in chapter 1 was *"What is an intelligent tutor system?"*. In chapter 3 I have clearly described that an ITS consists of a tutoring model, learner model and domain knowledge model, and explained the system should pass certain tests before it can be classified as intelligent.

In the background research chapter, I have also elaborated on several dynamic difficulty adjustment methods that were used later on in the conceptual models of the game. By observing the student, gathering statistics and using these to calculate how well the player has solved certain problems, we can estimate the skill level of the student and offer content tailored to that level the next time a request for new game content is made by the game client.

The last contextual problem is discussed in chapter 3, where we define the difficulty scale of the vocabulary content. By using pre-defined word lists that are labeled with significancy labels we can decide which content will be used in the game and what the easier and harder content will be.

9.2.3 Technical research answers

In chapter 4 we saw how we have elicited the requirements of an ITS to implement vocabulary expansion techniques. By keeping the EN closely in our feedback loop, we have seen that we can propose models based on known educational techniques and refine them using the feedback we gathered during meetings with experts.

In addition to the educational experts, all of the employees that are involved in the ITS were kept in the feedback loop. Through several presentations of the ITS during team meetings, everyone was able to provide feedback and express their desires and needs for the ITS.

Following the conceptual models, we created computational models based on the acquired information which are shown in chapter 5. By proposing, adjusting and refining we can create computational models that are still adjustable by including changeable variables, allowing us to fine-tune the system during the testing process.

The business plan of MniH always was to create an immersive game for young children and to extend the vocabulary knowledge while doing so. During the engineering of the ITS we have strived to produce optimal educational results which are needed to pass the real life tests mentioned in chapter 8 while keeping a close watch on the usability versus educational value tradeoff.

To integrate the ITS in the current game structure, several heavy rewriting iterations were needed to enable the system to handle the new and more structured educational content. Combined with the new script and scenario tool, a whole new workflow of educational content was added to the existing toolset. The new client server architecture written especially for the ITS system provides extra versatility and control over the content, which has a great effect on both the content of the game itself and the educational value of that content.

9.2.4 Internship results

In this final subsection I will give an overview of the results of the internship. First of all, the most important parts of the ITS including the server side logic and communication are fully implemented and provide an excellent base to further develop the MniH ITS.

For the parts that are not yet implemented due to time shortage, this document provides excellent programming guidelines. Using the architecture and pointers given in the implementation chapter it should be very feasible to implement and maintain the ITS so far.

Furthermore, a basic version of the design, conceptual models and architecture were used in an official report written by MniH for the Maatschappelijke Sectoren & ICT, which is a big honor for my work and makes a strong statement on the quality and significance of the work I have done in the development of the ITS.

The most important result however, is the experience and knowledge I have acquired in the field of intelligent tutoring and serious games in general. By fulfilling an important and responsible task during the development, I was given an opportunity to truly show my skills, knowledge and ability to adapt myself to an innovative field of expertise.

Bibliography

[1] Fabio N. Akhras and John A. Self. Beyond intelligent tutoring systems: Situations, interactions, processes and affordances. pages 1–30, 2002.

[2] Marios C. Angelides and Ray J. Paul. Towards a framework for integrating intelligent tutoring systems and gaming-simulation. In *WSC '93: Proceedings of the 25th conference on Winter simulation*, pages 1281–1289, New York, NY, USA, 1993. ACM.

[3] Robin Hunicke and Vernell Chapman. Ai for dynamic difficulty adjustment in games. 2004.

[4] R. Kaplan and D. Rock. New directions for intelligent tutoring. *AZ Expert*, page 10(2): 3040, 1995.

[5] M. Kienstra. *Woordenschatontwikkeling. Werkwijzen voor groep 1-4 van de basisschool*. Expertisecentrum Nederlands., 2003.

In this book, four vocabulary teaching prototypes are described that were developed by the Expertisecentrum Nederland

[6] Jean-Francois. Lelouche, Ruddy; Morin. Introduction of pedagogical concepts in domain modelling for an intelligent tutoring system. *European Journal of Engineering Education 23.2*, 1998.

This article presents knowledge modelling in an intelligent tutoring system, in which the modelling approach allows the knowledge representation to somewhat depart from its actual or most obvious basis to fit specifically pedagogical needs.

[7] M.L. Miller and S.R. Lucado. Integrating intelligent tutoring, computerbased training, and interactive video in a prototype maintenance trainer. *Intelligent Instruction by Computer*, pages 127–150, 1992.

[8] T. Murray. Authoring intelligent tutoring systems: An analysis of the state of the art. *International journal of artificial intelligence in education*, 10:98–129, 1999.

[9] Mijn naam is Haas. Mijn naam is haas, a serious game for young children, 2010.

[10] I. S. P. Nation. *Teaching and learning vocabulary*. Newbury House, 1990.

[11] Expertisecentrum Nederlands. Feedback in educatieve games en software: Een literatuurstudie naar de inzet van feedback in educatieve games ter bevordering van de taalontwikkeling van jonge kinderen, 2008.

This article, also specifically written for Mijn naam is Haas, is a scientifically substantiated literature study to feedback in educational games and software, focussing on the importance of feedback in language education and the implementation of feedback in educational games. In addition, the final chapter is devoted to the EN's recommendations and suggestions for the Mijn naam is Haas educational games content and interaction.

[12] Expertisecentrum Nederlands. Woordenschat een literatuurstudie naar de stimulering van woordenschatontwikkeling in een educatieve game, 2008.

This article, specifically written for Mijn naam is Haas, is a scientifically substantiated literature study to vocabulary expansion using known methods. It is written to give an insight into the methods used to stimulate vocabulary development and practical techniques that are used in modern education to improve this development. In addition, the final chapter is devoted to the EN's recommendations and suggestions for the Mijn naam is Haas educational games content and interaction.

[13] S. Ohlsson. Some principles of intelligent tutoring. pages 203–237, 1987.

[14] Timen P. Olthof. *Dynamic graphics in serious games*. CreateSpace, Amsterdam, 2010.

[15] Ehri L. C. Robbins C. Reading storybooks to kindergartners helps them learn new vocabulary words. *Journal of Educational Psychology*, 86 (1):54–64, 1994.

[16] Henk Pander Maat Rogier Kraf. Leesbaarheidsonderzoek: oude problemen, nieuwe kansen.

[17] Meagan K. Rothschild. The instructional design of an educational game: Form and function in jump, 2008.

This article on the design of the game JUMP gives an excellent example of an educational game that teaches words in a game using a feedback system that reacts on user behavior.

[18] Lejaegere Schaerlaekens, Kohnstamm. *Streeflijst woordenschat voor zesjarigen. Derde herziene versie gebaseerd op nieuw onderzoek in Nederland en Belgi*. Swets & Zeitlinger, 1999.

[19] Vermeer A. Schrooten W. *Woorden in het basisonderwijs. 15.000 woorden aangeboden aan leerlingen.* Tilburg University Press, 1994.

[20] Simon J. E. Taylor and Julika Siemer. Enhancing simulation education with intelligent tutoring systems. In *Proceedings of the 28th conference on Winter simulation*, WSC '96, pages 675–680, Washington, DC, USA, 1996. IEEE Computer Society.

[21] Dirkje van der Nulft en Marianne Verhallen. WOORDENSCHAT de 4-takt kwaliteitskaart, nov 2009.

This pamphlet describes the 4-takt vocabulary teaching method based on the face that words cannot be learned in one time and should thus be separated in 4 different steps of preparing, semantizing, consolidating and testing.

[22] Etienne Wenger. *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[23] Julie Wood. Can software support childrens vocabulary development. *Language Learning & Technology*, 5:166–201, 2001.

Appendices

APPENDIX A

API Backend Scenario's

A.1 login(loginparams) scenario

Call name	login
Actor	Game client
Summary	Login a teacher / parent to enter the administration panel
Scenario steps	1. Provide username and password and package these into login parameters package 2. Send the login parameters to the server 3. Server parses the parameters and queries the database to check the user credentials 4. An acknowledgement is sent back to the Game client with a status message that contains a message whether or not the credentials were accepted and a session cookie id which can be used in further steps to authenticate the web session 5. Game client receives Acknowledgement package and either opens the administrator panel or displays an error that wrong credentials were used
Assumption	Web application is connected to the internet, user credentials are known to the system
Exceptions	None
Result	User is logged in the web application and a session cookie id is returned which is used to authenticate the web session

A.2 matchSituation(situationparams) scenario

Call name	matchSituation
Actor	Game client
Summary	Load the available soil types for the current situation
Scenario steps	1. Create situation parameters package from the relief that was generated from the drawn line, the current stage type and the subject id 2. Send the situation parameters to the server 3. Server parses the parameters and queries the database for available soils for the given situation 4. Data is packaged into Soil object and sent back to the game client 5. Game client receives Soil object and uses that information to fill the drawn landscape with a soil
Assumption	Game client is connected to the internet, there are soil types available for the current situation
Exceptions	None
Result	Game is provided with a soil, that is used to fill the drawn landscape

A.3 loadStage(stageparams) scenario

Call name	loadStage
Actor	Game client
Summary	Load stage information including solutions, dialog and actions that is most suitable for the currently selected student based on his or her progression and skill
Scenario steps	1. Create stage parameters package from available data, containing the current student IDs, current relief, soil, excluded actors / stages, forced stage (optional), the current subject (game title), landscape, type (stagestep), concerns, cover and climate 2. Send the stage parameters to the server 3. Server parses the parameters and queries the database for solutions, dialogs and actions, based on the preferred word level of the student 4. Data is packaged into Stage object and sent back to the game client 5. Game client receives Stage object and loads the content into the engine
Assumption	Game client is connected to the internet, student is registered in the system
Exceptions	When a forced stage is specified then that specific stage is always loaded
Result	Game is provided with a new Stage, tailored to the currently playing student

A.4 sendStats(statsparams) scenario

Call name	sendStats
Actor	Game client
Summary	Send a prepared statistics package for one session to the game server and process the data
Scenario steps	1. Create statistics parameters package from gathered session statistics 2. Send the statistics parameters to the server 3. • When transmission times out, retry sending • When max number of retries is reached, queue parameters package to be sent to the server next time 4. When received successfully, send acknowledgement back to game client 5. The server parses the received package and updates the statistics for the involved student
Assumption	Game client is connected to the internet. If not, sending fails and package is queued for transmission
Exceptions	None
Result	User statistics for played session are processed: level completion, occurrence counts and contextualization count values are updated

APPENDIX B

Algorithm equations and variable values

B.1 Equations

B.1.1 Solution score calculation

$$s_1 = \text{MAX\_SCORE\_INSTRUCTIONS} - 0.1 * \text{instruction index} \tag{B.1.1.1}$$

$$s_2 = \text{MAX\_SCORE\_DRAWMATCH} * \left(\frac{\text{MAX\_DIFFERENCE} - \left(\frac{\sum_{i=1}^{n} (\text{bestmatch}_i - \text{matchscore}_i)}{\text{number of matches}} \right)}{\text{MAX\_DIFFERENCE}} \right) \tag{B.1.1.2}$$

$$s_3 = \text{\#contextualization sentences * MAX\_SCORE\_CONTEXTUALIZATION / 10} \tag{B.1.1.3}$$

$$\text{solutionscore} = s_1 + s_2 + s_3 \tag{B.1.1.4}$$

B.1.2 Dynamic Difficulty Adjustment

$$\text{average solution score} = \frac{\sum_{i=1}^{n} \text{solutionscore}_i}{n} \tag{B.1.2.1}$$

$$\text{solution score difference} = \text{average solutionscore} - \text{solutionscore} \tag{B.1.2.2}$$

$$\text{average mismatch} = \frac{\sum_{i=1}^{n} (\text{pref. wordlevel} - \text{stageplugin difficulty})}{n} \tag{B.1.2.3}$$

$$\text{weighted frame average} = \sum_{i=1}^{n} \text{solution score difference}_i * \text{weighting}_{\text{frameIndex}} \tag{B.1.2.4}$$

$$\text{LOWER\_BOUND} < \text{weighted frame average} < \text{HIGHER\_BOUND} \tag{B.1.2.5}$$

B.1.3 Priority calculation

$$\text{priority} = \text{LOWER\_PRIORITY\_BASE}^{(1+\text{number of occurrences} - \text{MAX\_OCCURRENCE})} \tag{B.1.3.1}$$

$$\text{priority delta} = \left(\frac{\text{HIGH\_PRIORITY} - \text{NORMAL\_PRIORITY}}{\text{target word occurrence} - \text{MIN\_OCCURRENCE}} \right) \tag{B.1.3.2}$$

B.2 Variable values

`MAX_SCORE_INSTRUCTIONS` - Maximum score for first part of solution score calculation (Default: 1)

`MAX_SCORE_DRAWMATCH` - Maximum score for second part of solution score calculation (Default: 0.5)

`MAX_SCORE_CONTEXTUALIZATION` - Maximum score for third part of solution score calculation (Default: 0.1)

`INSTRUCTION_PENALTY` - The amount of solution score subtracted for each instruction index (Default: 0.1)

`INSTRUCTION_INDEX_BEFORE_PENALTY` - The amount of instructions before a penalty is applied (Default: 3)

`MAX_DIFFERENCE` - Maximum amount of match score difference between drawn object and correct solution object (Default: 20)

`STUDENT_AVERAGE_COUNT` - Amount of students used for average solution score value calculation (Default: 50)

`FRAME_SIZE` - Amount of previous student solution score difference averages taken into account (Default: 5)

`FRAME_WEIGHTS` - Values determining the weight of first, second, etc. frame value (Default: "0.6,0.2,0.1,0.075,0.025")

`FRAME_AVERAGE_LOWER_BOUND` - Lower bound of student performance "comfort zone" (Default: -0.2)

`FRAME_AVERAGE_UPPER_BOUND` - Upper bound of student performance "comfort zone" (Default: -0.1)

`INCREASE_RATE` - Amount of levels increased when leveling up during dynamic difficulty adjustment (Default: 0.25)

`DECREASE_RATE` - Amount of levels decreased when leveling down dynamic difficulty adjustment (Default: 3)

`MIN_OCCURRENCE` - Minimal number of occurrences for a certain target word before lowering the priority (Default: 2)

`MAX_OCCURRENCE` - Maximal number of occurrences for a certain target word before lowering the priority (Default: 3)

`HIGH_PRIORITY` - High priority value for words that have to be repeated as soon as possible (Default: 2)

`NORMAL_PRIORITY` - Normal priority for words that have not occurred yet (Default: 1)

`BELOW_AVERAGE` - If a solution score difference is far below average, the priority for that target word stays high. (Default: -0.2)

`LOWER_PRIORITY_BASE` - Value for the base of the exponentiation in the priority calculation (Default: 0.9)

`MIN_WORDLEVEL` - The minimum possible word level, could be used to change the bounds of the available word levels (Default: 1)

`MAX_WORDLEVEL` - The maximum possible word level, could be used to change the bounds of the available word levels (Default: 100)

`BIG_MISMATCH` - Defines when a preferred word level and plugin word level mismatch is considered big enough to react (Default: 40)

APPENDIX C

Database architecture diagram

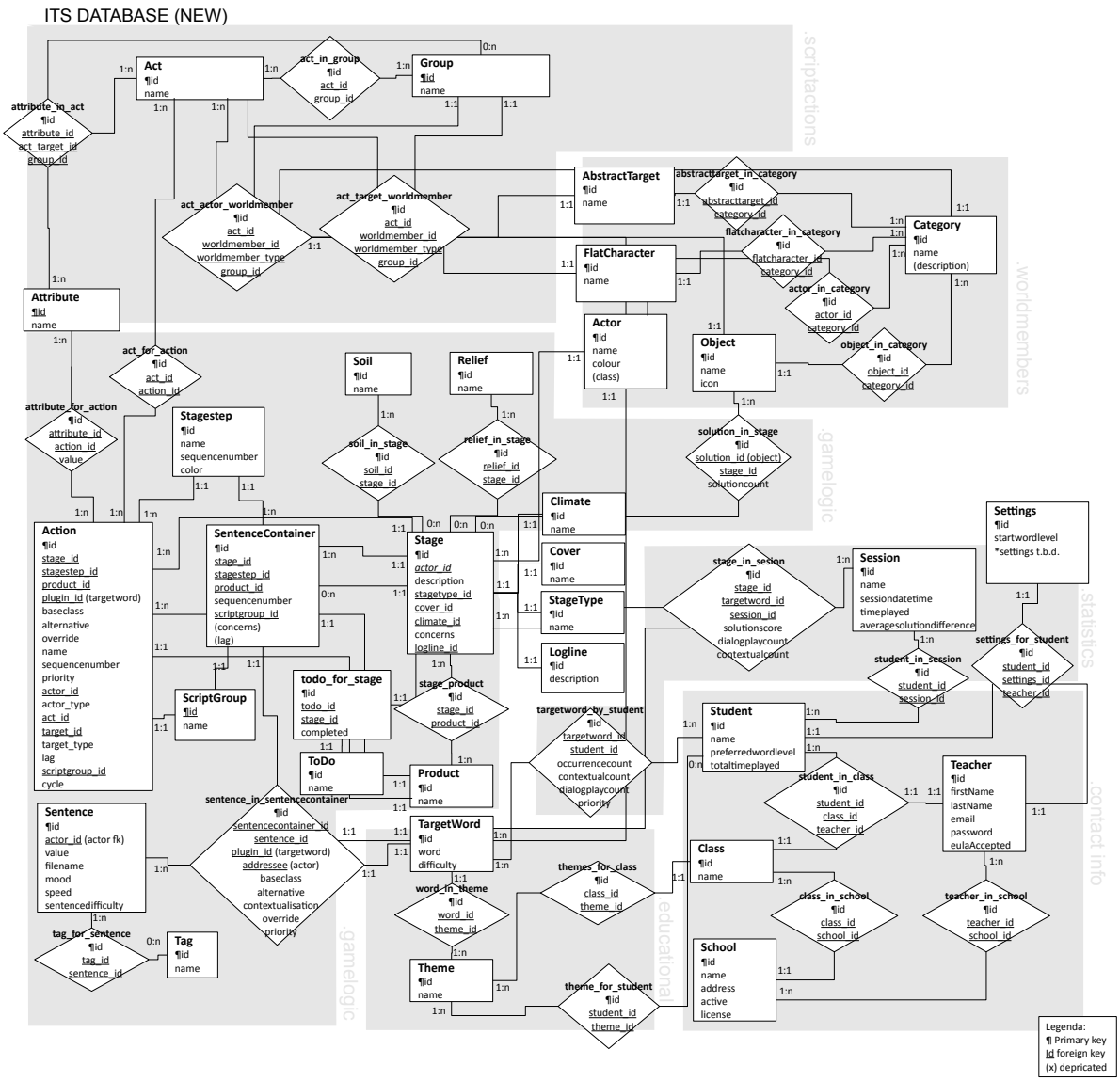

Figure C.1: ITS enabled MniH relational database.