Teaching children by computer: an educational software architecture for the seesaw problem

Bachelor thesis
Marjolein Terhaar
s0105848
Supervisor: dr. P.A. Kamsteeg
Academiejaar 2007–2008
Abstract

In this thesis an educational computer system is described to teach the physics phenomenon of equilibrium to children. Normally this domain is considered to be too hard for children to understand. To investigate if this can be overcome by making the domain more concrete, an educational software system has to be built. Therefore the following two question had to be answered: what kind of environment to use and how to make the concepts more concrete. It has been chosen to use a discovery environment and the domain concepts had been made more concrete by using a simulated seesaw and a deviation of the educational material into different levels. To implement the educational design a general object-oriented framework has been built. With this framework it is possible to build the educational system and test the research question.
# Contents

Abstract i

1 Introduction 1
   1.1 The importance of educational computer systems 1
   1.2 The interesting aspects of educational computer systems 1
       1.2.1 The educational point of view 1
       1.2.2 The artificial intelligence point of view 2
   1.3 Subject of the thesis 2
   1.4 Structure of the thesis 3

2 Theoretical background and problem statement 4
   2.1 Educational software 4
       2.1.1 History 4
       2.1.2 Types of educational systems 4
   2.2 Theory of education 5
       2.2.1 Cognitive development of children 6
       2.2.2 The need for stimulating learning environments 7
   2.3 Research question 7

3 Educational design 8
   3.1 Introduction 8
       3.1.1 The complexity of mental models 8
   3.2 What the child should learn 11
   3.3 Requirements of the system 11

4 Architectural framework 13
   4.1 Software design 13
       4.1.1 What kind of architectures 13
       4.1.2 An object oriented design 15
   4.2 The resulting software framework 15

5 Conclusion 22

Appendix A Objects 24

References 33
Chapter 1

Introduction

This thesis is about educational computer systems. These are systems which communicate knowledge of some domain to the learner by the use of a computer. The communicated knowledge can be of different kinds, for example facts about the domain, the understanding of some process, or the skill to perform certain actions. So instead of traditional education where a human teacher communicates the domain knowledge to the student or where a student learns from a book, the computer will (partly) take over this task.

Educational computer systems may be adaptive to some extent, that is, they can adapt their educational actions to the current needs of a particular student. They could for instance give extra explanations or exercises on topics that the student does not yet master.

1.1 The importance of educational computer systems

Educational computer systems are important because they allow a number of possibilities that are hard or impossible to achieve in other ways of teaching. First of all, they provide a way to give one-on-one education without the need for a human teacher. Moreover, they make it possible for the student to work at his own speed and to repeat the material as often as he needs. And they make it more feasible to present the material in different modalities such as for example video, audio or simulation. In addition, some ways of presenting are exclusively possible with a computer, for example simulations. Simulations make it possible for the student to gain experiences without being exposed to danger.

1.2 The interesting aspects of educational computer systems

1.2.1 The educational point of view

Educational computer systems pose the interesting question of ‘how to teach’. Yet the only model we can use as an example of this are human teachers. But as Ohlsson (1986) states we still do not exactly know how human teachers teach. So he proposes to first investigate what methods the teachers use, before we decide which methods we can apply to computer systems and which methods we should adjust or specially design for educational computer systems.
CHAPTER 1. INTRODUCTION

Educational systems for children

Educational systems for children are especially interesting because of the call for edutainment. Edutainment is a composition of education and entertainment. And with regard to educational systems for children, both are important, because children do not have a very long attention span and may not be motivated to gain knowledge on their own.

Another interesting aspect is the question whether it is possible to make knowledge that is deemed to be too advanced for young children accessible for them, by adapting the knowledge to their own cognitive level. More about this in chapter 2.

1.2.2 The artificial intelligence point of view

From an artificial intelligence point of view educational systems present difficult problems to solve. For example, one of the challenges is how to make the educational system optimally adaptive. To adapt, the system must be able to reason about each specific student and make decisions about which teaching material to use and how to present it to the him. This requires the development and maintenance of a model of the student, which raises the questions of what that model should contain so that the system can reason about the student, and how this information should be updated. Diagnoses of the mistakes the student makes are needed, so that the model of the student can be updated with the correct and incorrect knowledge of the student.

The subject that is being taught, the domain, should be modeled as well, and this is also a problem. How to represent this domain knowledge? The knowledge of the domain must be adequately represented to be able to reason about the domain. And because the domain knowledge is used to teach the domain, there are extra constraints on this knowledge.

Another problem to be solved is how to generate assignments. What difficulty should the assignment have, and how to generate assignments of a certain difficulty level? And how can the assignments be built from the domain model? How can the system ensure that each assignment is different, and in what way should these assignments differ? Yet another important question is how to solve these assignments. The system must be capable of solving the assignments because it must check if the solution presented by the student is correct. It may even be required that the system is capable of solving assignments in several ways, if different correct solutions are in fact possible. These questions could be answered by intelligent solutions provided by the domain of artificial intelligence.

But before we can look at intelligent solutions, there has to be an educational design.

1.3 Subject of the thesis

The specific subject of this thesis is an educational system for children at the higher levels of elementary school. In this system children will learn about a domain which is generally assumed to be above their cognitive level, namely the domain of physical equilibrium. This domain has been chosen as an example of abstract physical laws. The questions to be answered will be: Which elements need to be incorporated in an educa-
tional system in order to teach 11/12 year old children about the equilibrium problem? How can such a system be build?

1.4 Structure of the thesis

The thesis consists of the following parts. First in Chapter 2 I will present some background literature about educational software and education in general, and I will present the research question. Then in Chapter 3 I will present the educational design for the physical equilibrium problem. In Chapter 4 I will discuss how to translate this educational design into a software architecture. In Chapter 5 I will present the conclusions of this thesis. And in the appendix I will present the remaining objects that do not belong to the main objects presented in Chapter 4.
Chapter 2

Theoretical background and problem statement

2.1 Educational software

2.1.1 History

The development of education by computers started around 1963 when John Kemeny and Thomas Kurtz developed the easy to use programming language BASIC, which made it feasible to create computer-based instructions without specialist computer skills (Molnar, 1997).

The earliest or at least most primitive software for education was based on linear mastery learning. These systems kept on repeating the material until the student mastered it. But these systems where only useable for pure “learning by heart” and were not very effective for other learning methods (Molnar, 1997).

In reaction to this shortcoming, Computer-Aided Instruction (CAI) systems were developed, which used a branching method to make small adaptions to the succeeding material based on the answers of the student. These systems responded in distinct ways to the answers of the student, following the different branches in the decision tree. But this method gave only limited opportunities to adapt the behaviour of the system to the student, because the system could only react on a small set of expected answers from the student (Wenger, 1987).

In reaction to this, Intelligent Computer-Aided Instruction (ICAI) systems were developed. These systems use localized intelligence to analyze the student’s performance and adapt the learning proces to the student. Small parts of the system are made intelligent, for example the part that generates assignments. But, just as in CAI systems, the remaining part of the knowledge remains in the designers head, and only the decisions are programmed. In an ideal system all the knowledge should be represented in the system, this way making it possible for the system to reason about this knowledge and communicate this knowledge to the student. Such a system is called an Intelligent Tutoring System (ITS) (Wenger, 1987).

2.1.2 Types of educational systems

There are different types of educational computer systems just as there are different viewpoints on education. A way to categorize the different educational systems is by looking at
the different goals of education. These goals can be things like learning facts about a domain, understanding the concepts that play a role in a domain, learning actions skills, or mental skills such as problem solving. There are different views in education about the best methods to reach these goals.

For instance, a generally accepted method for learning facts about a domain is by repeating the material over and over again. This method can be very effective to use when students have to learn translations of words. However, a different viewpoint is that even simple facts can be learned better by embedding those facts in actual situations of problems to be solved. Another view is that teaching by giving an explanation is effective when learning concepts about a domain. Just repeating over and over the concepts will be inadequate for remembering and especially understanding these concepts. When given an explanation, the material will be understood much better and therefore presumably also better remembered. A drawback of this way of teaching is that the student is very passive, he just has to listen and the teacher can not be certain that the student really understands the material.

A next learning method is learning by practicing skills, which is effective in learning action skills such as riding a bicycle. Just telling someone how to ride a bicycle is not very effective, the student has to practice this skill. But also for mental skills practice is needed. When teaching mental skills such as problem solving, some explanation is needed, but practice is also required to master these skills.

And yet another learning method is discovery learning (Bruner, 1961), where students can interact with their environment and possibly perform experiments in order to learn the material. The idea is that students will remember the material better when they have discovered the material on their own, because the student is forced to actively participate in the learning process.

In ITSs all these different ways of teaching are used, for example coaching systems, which are usually used to teach problem solving, or simulation environments where students can practice their skills in a safe environment. Reification, making things visible that are not visible in the real world, plays an important role in simulation environments. For example in Steamer (Wenger, 1987) the inside of a steam plant has been made visible. This way it is easier for the student to see and understand fully what happens when operating the steam plant.

Video games are a recent development in ITSs. One of the advantages of video games is that they are very appealing to children (Squire, 2002).

Other recent developments are the use of multimedia (Okamoto, Cristea, & Kayama, 2001) and groupware systems (Alavi, 1994). Multimedia is used to build more realistic virtual environments, to make the software more attractive or to provide several alternative ways to elucidate some phenomenon. In groupware systems users work together on solutions to problems and therefore can help each other and learn from each other. The computer may play the role of one co-learner in such a group and/or of a general advisor/supervisor for the group.

2.2 Theory of education

Because a main theme of this thesis is the cognitive capabilities of small children and the question if their capabilities can be extended, I will first discuss Piaget and his research into
the cognitive capabilities of children and then Papert (1980) with his vision on education.

2.2.1 Cognitive development of children

The cognitive development of children has been studied extensively by Jean Piaget (Flavell, Miller, & Miller, 2002). Piaget describes four consecutive stages in the cognitive development of children. These stages are

- Sensorimotor
- Preoperational
- Concrete-operational
- Formal-operational

The sensorimotor stage occurs between birth and approximately 2 years. In this stage children understand the world by acting on it, they form mental representations of the world by physical interaction with the world. At the end of this period children can form mental representations of the reality, but only in terms of actual physical interactions.

The preoperational stage occurs at the age of 2 to 7 years old. In this stage children are capable of using representations such as words and drawings to think about objects and events. But these children are not yet capable to think about abstract concepts, such as “negative number”.

The next stage, the concrete-operational stage occurs at the age of 7 to 11. At this stage children gradually learn to think in an abstract way and learn how to use concepts. Children learn to make logical connections with the things they already master.

The last stage is the formal-operational stage, occurring at the age 11 to 15. This stage is known for the capability to perform mental operations on both possible and hypothetical situations. Children at this stage learn scientific hypothetico-deductive reasoning.

The idea of Piaget was that children are not capable of performing cognitive actions that belong to a higher stage. This would mean that children at elementary school, who are in the concrete-operational stage, are not capable of performing hypothetico-deductive reasoning and have trouble understanding the process of physical equilibrium.

The idea of Piaget (Flavell et al., 2002) is that children develop their mental skills by assimilation and accommodation. Assimilation is the process of the child applying his present knowledge to the world around him. Accommodation is the opposite; adjusting his knowledge to the characteristics of the world around him. It is an interaction of interpreting the world and adapting the mental model of the world. First the child tries to fit the world into his mental model, but when he learns more and more about the world, this is not sufficient and he will have to change his mental model to adapt to the world. In this way the brain of the child changes over time and the cognitive ability of the child grows (Flavell et al., 2002). Everyday experiences underlie the cognitive development of children.
2.2.2 The need for stimulating learning environments

Papert (1980) also stresses the importance of experience in the development of children. Papert builds on the ideas of Piaget. He agrees with Piaget on the idea of accommodation and assimilation and the important role of experiences, but he disagrees with the ideas of Piaget regarding the different stages of cognitive development. Papert states, in contrast to Piaget, that it is possible for children to learn things that belong to a higher stage than the child is actually in. That is why Papert has developed the simplified computer language LOGO, to help young children discover abstract concepts (Papert, 1980).

Papert also developed the theory of constructivism. The main idea of constructivism (Genalo, Schmidt, & Schiltz, 2004) is mental construction, building knowledge structures about the things in the world. This works best for children when they are able to explore and discover solutions to problems in (simulated) realistic learning environments.

This idea is very similar to the idea of Bruner (1961) of discovery learning, which has already be mentioned as a method of education. A drawback of this kind of learning is that the student has to form his own hypotheses and analyse the results of the experiments on his own. When the student is not capable of this, the student will learn nothing or even wrong things. That is why it is important to guide the student when performing experiments.

2.3 Research question

The question I am interested in is if it is possible to teach children about a domain that is generally assumed to be too difficult for them. Are these domains just to hard to get for children, no matter how you present the domain, because the domain asks for cognitive actions that belong to a higher stage than the stage of the child, as Piaget (Flavell et al., 2002) would state? Or is it possible, by presenting the domain at a lower cognitive level, for children to understand the domain, as Papert (Papert, 1980) would suggest.

I have chosen to test this question on the domain of physical equilibrium, because this problem requires formal-reasoning and according to Piaget, children at the higher classes of elementary school are not yet capable of doing this.

So the research question is: if the domain of physical equilibrium is presented at a more concrete way, will children at the higher classes of elementary school understand the concepts of the domain?

In order to answer this question an educational software environment will be build, this way the presentation of the problem, and thereby the cognitive level, can be adjusted for each child. In order to build such an environment the following two questions have to be answered. What kind of educational environment to use. And second, how the abstract laws of physical equilibrium can presented in an concrete way.
Chapter 3

Educational design

3.1 Introduction

To answer the research question, first an educational design will be developed. In this design it will be argued what kind of education software environment to use and it will be demonstrated how to make the abstract concepts of the equilibrium domain more concrete.

For the educational software environment it has been chosen to use a discovery environment where children are able to experiment on their own with the equilibrium problem. Bruner (1961) argues that children will remember the concepts better by discovering them on their own.

In order to make the domain more concrete the following two decisions have been made. The first decision is to use a seesaw to make the principles of physical equilibrium clear. I have opted for a seesaw because children are very familiar with seesaws and it is a tangible device where the two arms and the fulcrum are clear to see and simple to use.

The second decision is to split the educational material into different levels, so the child will be able to construct and extend his mental model in a gradual and systematic way. The levels that will be used follow the division described by Frederiksen and White (1988). In the following section this division will be described.

3.1.1 The complexity of mental models

Frederiksen and White (1988) describe two dimensions on which mental models may vary in complexity. They call them the order of a model and the degree of elaboration of a model. The order of a model says something about the type of variables that are used in reasoning. In a zero order model boolean variables are used, in the case of the seesaw “equilibrium or no equilibrium”.

First order models are models that reason about the amount of something, these models can answer questions of the form “is there more, equal or less of something”. Or in the case of the seesaw, “to what side will the seesaw tilt” due to the more, less or equal mechanical advantages of the two arms.

Second order models can answer questions about the change of things, such as questions of the form: “which seesaw tilts faster”.

8


<table>
<thead>
<tr>
<th>Level</th>
<th>Degree of Elaboration</th>
<th>Order of the Model</th>
<th>Complexity of Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>first degree; only the parameter mass is adjustable</td>
<td>zero order</td>
<td>compare amount of blocks for both sides</td>
</tr>
<tr>
<td>2</td>
<td>&quot; &quot;</td>
<td>first order</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot; &quot;</td>
<td>second order</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>4</td>
<td>first degree; only the parameter distance is adjustable</td>
<td>zero order</td>
<td>compare the distance of the blocks to the fulcrum for both sides</td>
</tr>
<tr>
<td>5</td>
<td>&quot; &quot;</td>
<td>first order</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot; &quot;</td>
<td>second order</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>7</td>
<td>second degree; both parameters mass and distance are adjustable</td>
<td>zero order</td>
<td>comparing the moment of both sides; single calculation of moment for both sides</td>
</tr>
<tr>
<td></td>
<td>one pile of mass on each side allowed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>&quot; &quot;</td>
<td>first order</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>9</td>
<td>&quot; &quot;</td>
<td>second order</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>10</td>
<td>second degree; both parameters mass and distance are adjustable</td>
<td>zero order</td>
<td>comparing the moment of both sides; moment of one side is the sum over the moments of each pile on that side</td>
</tr>
<tr>
<td></td>
<td>multiple piles of mass on each side allowed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>&quot; &quot;</td>
<td>first order</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>12</td>
<td>&quot; &quot;</td>
<td>second order</td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

Figure 3.1: Overview of the different levels of the education system. Each level differs in the complexity of the model by varying the degree of elaboration, the order of the model and the complexity of calculation.
The degree of elaboration of a model says something about the number of component types and therefore the amount of parameters that play a role (Frederiksen & White, 1988). The more parameters play a role in the model, the higher the degree of elaboration. In the domain of a seesaw only two parameters are important, namely the masses that are put on the seesaw and the distance of the masses to the fulcrum. The first degree of elaboration is created when only one parameter can be adjusted, so only the mass or only the distance. A higher degree of elaboration is created when both mass and distance can be adjusted.

Within the first degree of elaboration the child only has to compare the amount of blocks on both sides of the seesaw, or the distance of the blocks to the fulcrum, to determine if the seesaw will tilt, to what side and how fast it will tilt. Within the second degree of elaboration the moment of each side has to be calculated and these moments have to be compared to determine what the seesaw will do. Within this second degree of elaboration we can adjust the complexity of calculation that has been done (Frederiksen & White, 1988). The first level of complexity consist of the possibility to adjust both the mass and the distance to the fulcrum, but all the mass has to be piled up on each other. So there is only one pile of masses which can be placed on a chosen distance from the fulcrum. To calculate the moment for one arm, the amount of masses on that arm has to be multiplied by the distance of the masses to the fulcrum. A second level of complexity also consist of the possibility to adjust both the mass and the distance to the fulcrum, but now multiple piles on each arm are allowed. These piles can be placed at multiple chosen distances to the fulcrum. So the calculation gets a little bit harder this time, because now to calculate the moment of one arm, every pile of masses on that arm has to be multiplied by the distance to the fulcrum of that pile. The calculations for all these piles have to be added up to each other to get the resulting moment for that arm. The second complexity level asks for more calculations than the first complexity level.

So we have the possibility to create 4 different situations with the two variables. Two situations where only one parameter can be adjusted. And for the higher degree of elaboration, also two complexity levels where both parameters can be adjusted.

We can use this division to create the different levels. Frederiksen and White (1988) describe a way to impose a sequence of these levels for the curriculum. They keep the degree of elaboration the same and increase the order of the model for each level. See Table 3.1 for the resulting twelve levels.

The first level starts with the first degree of elaboration. It has been chosen to first make the mass adjustable and keep the distance to the fulcrum constant for both sides of the seesaw, because intuitively the effect of the amount of mass seems easier to understand than the effect of distance to the fulcrum. The order of the model is increased until the second order has been reached. Subsequently in the fourth level the mass will be kept constant and the distance to the fulcrum can be adjusted by the child. The fourth level consists of the zero order and this order is increased for each next level until the second order. In the seventh level the second degree of elaboration and the zero order is used. From this level on the first complexity of calculation is used. And for each subsequent level the order is increased.
In the tenth level also the second degree of elaboration and the zero order are used, and this time the second complexity level for the calculation of the moment is used. And again for each subsequent level the order is increased.

3.2 What the child should learn

The purpose of this system is to teach children the principles concerning physical equilibrium. Therefore they must understand the phenomena balance and imbalance, what makes a seesaw tilt to a specific side and what makes a seesaw tilt faster or slower to a side. So the children must be capable to predict when there is a balance and imbalance, they must be able to predict to what side a seesaw will tilt and predict when a seesaw will tilt faster. In this way they will learn to understand the influence of the variables weight and distance to the fulcrum. But in order to do so, they should also learn to perform good experiments and learn to draw the correct conclusions from these experiments.

Guidance for the experiments and feedback on the child’s answers will be given by the system. The system must monitor the student when he is experimenting and should give guidance when the material is not well understood or when wrong experiments are performed.

3.3 Requirements of the system

The system must fulfill certain requirements with respect to the way the system should function. I will first discuss what kind of functionality is needed from the system and then present these requirements.

The child must be able to login to the system, so that a personal record for that child can be made. This way the system can adapt itself to that child and when the child logs in a next time, the child can continue from where he has left.

As I have chosen for a discovery learning environment, the child must be able to experiment with the seesaw. To make the actions as concrete as possible I have chosen to use an animation of a seesaw. The child can drag blocks onto this seesaw with his mouse.

The parameters distance to the fulcrum and mass placed on the seesaw should be standardized to make it more clear to the child what the effect of the parameters is. The blocks should be standardized by using blocks of the same material, size and therefore weight. This way the child has to pay attention to the amount of blocks only and not to the different weights of the blocks.

The distance to the fulcrum should also be standardized by making fixed spots on equal distances of each other where the blocks can be placed. The blocks can only be placed on these spots, limiting the possible distances to the fulcrum to fixed and easy to discriminate ones. This way it is easy for example to see that some block is placed two times further away from the fulcrum than some other block.

The child should choose between different fixed answer possibilities for each level. In this way it is easy for the system to check if the answer was correct. And besides that, with fixed answer possibilities the child will be forced to give answers of increasing insight into the problem, because also the order of the model increases.
Because the child is free to experiment for himself, feedback on this process is very important, so the system must be able to follow the child during experimentation and must be able to give the child feedback on experimentation. The first kind of feedback the child should get is whether his prediction was correct or not correct. The second kind is feedback on the way the child performs the experiments. When the child does not vary enough between the different experiments, or when the child tries to do things that are not allowed, the child should be given a hint on experimentation. The third kind of feedback involves hints on the domain. For example if the child makes the same mistake a couple of times, the child should be given a hint to get some help on that part of the domain. This help section is needed as background information on the domain. The child must be able to search for some help on the subjects he does not understand.

The following functionality is needed:

- login ability for the system
- to use a simulated device where the child can experiment with
- standardize the parameters of the domain
- using fixed answer possibilities
- the system must follow the child and give the child feedback about the experiments and the domain
Chapter 4

Architectural framework

Now that we have the educational design, the next step is to decide how to translate this design into a software architecture. The first idea was to use an already existing simulation package. At first instance the SimQuest environment (Joolingen, King, & Jong, 1997) seemed suitable, but eventually not everything that was needed according to the education design was possible in this environment, such as login to the system and keeping a record of the child. It was also not possible for the child to freely experiment, because in SimQuest the required settings for each experiment had to be given in advance by the maker of the program. Since I did not found any other packages suitable for my design, I needed to build one myself. I have chosen to build a general framework that could also be used for problems similar to the seesaw problem, because this would save a lot of effort the next time. However, this decision implied that actual specific design and implementation exceeded the scope of this thesis.

In the next section I will describe the design decisions I have made, followed by the description of the resulting framework and the possible intelligent parts that could fit into this framework.

4.1 Software design

4.1.1 What kind of architectures

First I will discuss the general architecture to be used for the framework. The architecture describes the main building blocks, their functions and their interactions.

According to Wenger (1987) ITS’s can be divided into four components, the domain knowledge, student model, pedagogical knowledge and the interface.

The domain knowledge consists of the knowledge about the domain that is being taught and that the student should learn.

The student model consists of the necessary information about the student. The student model should contain information about the student that is needed for the system to be able to adapt its behaviour to the student.

The pedagogical knowledge consists of the knowledge about teaching, and concerns for example how to teach the material in a clear way.

The interface takes care of the communication between the human and the computer.

These components should be seen as more or less self-contained experts that cooperate in constructing the educational process.
Winkels and Breuker (1992) state that this decomposition into components shows too much overlap. For example the knowledge that a particular concept has to be understood before another concepts can be understood, is both domain and pedagogical knowledge. This is one reason why Winkels and Breuker (1992) argue for a decomposition into functions. Another reason is that functions are more general and can be reused more easily.

The functions that Winkels and Breuker use are planning, monitoring and diagnosing (Figure 4.1). The loop of planning an action, monitoring, diagnosing, and re-planning of the process takes place on three different levels of process control, the curriculum level, the task level and the communication level. When an action is planned on a higher level, this is communicated to the level below and here the action will be split up in different actions, which are again being planned, monitored, diagnosed and re-planned on this level.

Winkels and Breuker (1992) distinguish between the functions and the data. The data consists of the student model and the domain knowlegde. The functions can communicate with the data, but these data play no active role in the process control.

![Figure 4.1: Three levels of process control in an ITS. Every level has three functions: P (planning), M (monitoring) and D (diagnosing), around Pf (performance). From Winkels & Breuker(1992).](image)

The difference between the architecture of Wenger (1987) and Winkels and Breuker (1992) is that the architecture of Wenger is based on “who does it”, so the different “persons” will perform the different task, while Winkels and Breuker use a division based on “what needs to be done”.

The advantage of the division according to Winkels and Breuker (1992) is that the different parts are clear and coherent parts, with their own autonomous tasks. This makes it possible to design the different parts separately from each other, while in the design of Wenger (1987) the components overlap. The components proposed by Winkels and Breuker (1992) are domain independent, which makes them reusable for other domains. For these reasons I have chosen for the functional decomposition based on Winkels and Breuker (1992).
4.1.2 An object oriented design

Object oriented design is based on a decomposition of domains in terms of objects instead of functions as in some other design paradigms. Objects are active self-contained entities that play a particular role in a domain and have responsibilities in relation to that role. The advantage of an object-oriented design is that changes in the requirements of the program can be more easily made. The objects are largely self-contained and changing an object would therefore have minimal impact on the other objects.

At first instance an object oriented design may appear to be better suited for a structural division as used by Wenger (1987), because object-oriented design uses a decomposition into objects instead of functions. But the functions that Winkels and Breuker (1992) describe are parts with their own responsibilities, to the contrary of the components of Wenger (1987), which contain overlap between the components. And besides that, objects can also be purely functional.

In the next section the different objects and their tasks will be described.

4.2 The resulting software framework

The resulting object oriented framework is based on the educational design of chapter 3 and the architecture of Winkels and Breuker (1992). The different levels of process control in the framework resemble the levels described by Winkels and Breuker (1992). The objects in the framework that take care of the different levels of process control are ‘curriculum’, which takes care of the process control on the curriculum level, and ‘assignment’ which takes care of the process control on the task level. The ‘interface’ takes care of the process control on the communication level. As in Winkels and Breuker (1992) ‘domain’ and ‘student model’ are distinct parts that are passive, they have no direct active role in the actual process control of the teaching.

The difference between this framework and that of Winkels and Breuker (1992) is the extra main object ‘session’, which takes care of the start and ending of a teaching session with the child.

The main objects are described below, together with a short indication of the secondary helpers they cooperate with. The secondary helpers are fully described in the appendix. An overview of the relations between the main objects is given in Figure 4.2.

Domain

The domain object (Figure 4.3) is a distinct object of ‘information provider’ type. An ‘information provider’ only stores and manages information and provides it on the request of other objects. It has no active role in the process control.

The domain consists of the ‘formal domain description’, which are the variables and laws of the domain, and the ‘assignment template’ which is a ready made template suitable for that domain, where only the variables and different objects have to be filled in.

The ‘domain’ has a name, so that the right domain can be read in by the ‘session’.

The responsibilities of the domain object are:

- knowing the name of the domain
Student model

The ‘student’ object (Figure 4.4) is also an ‘information provider’ object that has no active role in the process control. All the important information that is needed for the system to adapt itself to the student is stored in the ‘student knowledge’. This knowledge can contain simple information such as the settings of the experiment, the prediction of the student and if this prediction was right. But this knowledge could also contain other information such as specific mistakes the student often makes and the underlying misconceptions. The ‘student’ object contains the login name of a student, his learning style, the level number and the domain the student is working on. The responsibilities of the ‘student’ object are:

- knowing the login name of the student
- knowing the learning style of the student
- knowing the knowledge of the student
- knowing level number of the student
- knowing the name of the domain the student is working on
- knowing other relevant details such as special mistakes, misconceptions and pre-knowledge of the domain
 CHAPTER 4. ARCHITECTURAL FRAMEWORK

The ‘session’ object (Figure 4.5) is an object that initializes and ends a session with a student. When a student logs in, his login name is used to load the proper information about that him from his student file to the ‘student model’. Also the domain the student should work on is loaded from the domain file. During each session the teaching of the domain is delegated to the ‘curriculum’. When the student logs out, the updated student model is written back into the student file. This information may be as simple as a level number indicating the part of the curriculum the student has fulfilled, but it could contain any type and amount of student information. The responsibilities of the ‘session’ object are:

- load information about the student from the student file to the student model
- load the relevant domain from file
- delegate the task of teaching the domain to the ‘curriculum’
- store information about the student in his student file

The ‘curriculum’ takes care of the process control on the curriculum level (Figure 4.6). The ‘curriculum’ contains the functions plan, monitor and diagnose. The ‘curriculum’ plans the whole curriculum, which in the simplest form can be just loading the sequence of levels from a file. In a more intelligent way, the curriculum could decide on the basis of different parts of information, such as the learning style of the student and the knowledge of the student, what the best sequence of levels would be. It is also possible to make the ‘curriculum’ analyze for all the students that work on the system, which misconceptions are made often, or which parts of the domain are hard to understand, so that the ‘curriculum’ can give some extra
attention to these parts and misconceptions standard to each student.
The ‘curriculum’ monitors the learning process of the student, diagnosing possible causes of insufficient progress, re-planning the curriculum if necessary.
The task of process control for the assignments will be delegated to the ‘assignment’. The responsibilities of the ‘curriculum’ object are:

- plan or re-plan the sequence of levels
- delegate the task of process control for the assignments to ‘assignment’
- monitor, diagnose and re-plan the curriculum process of the student

Assignment

The ‘assignment’ (Figure 4.7) takes care of the process control on the task level. It plans a new assignment and delegates the task of creating the assignment to the helper ‘assignment creator’. The ‘assignment’ has also a helper function ‘assistant tutor’ which monitors the student during the experiment. The ‘assignment’ uses the ‘prediction and outcome comparer’ to determine if the prediction of the student was correct. The intelligent part ‘diagnose mistake’ could be used to reason about the underlying misconceptions or lack of knowledge of the student when the student has made a mistake.
In its simplest form the ‘assignment’ updates the ‘student model’ by storing the right and wrong predictions of the student, combined with the settings of the experiment and the prediction of the student. But this knowledge could also consist of the mistakes and the underlying misconceptions of the student if available from ‘diagnose mistake’.
The ‘assignment’ gives feedback on the prediction the student has made. The ‘assignment’ uses the ‘assignment display’ to show itself on screen. The responsibilities of the ‘assignment’ object are:
• plan a new assignment
• determine specifications for the assignment
• delegate the task of creating a new assignment from provided specification to ‘assignment creator’
• show itself on screen
• delegate the task of monitoring the student to ‘assistant tutor’
• delegate the task of determining if the prediction of the student was right to ‘prediction and outcome comparer’
• delegate the task of diagnosing the reason for the mistake of the student to ‘diagnose mistake’
• give the student feedback: tell if prediction was right or wrong and possible feedback on the mistake
• update the ‘student knowledge’
• re-plan the assignment in case the prediction of the student was wrong

**Interface**
The ‘interface’ object (Figure 4.8) consists of an ‘output’ object which takes care of the output of the system to the student, and an ‘input’ objects which takes care of the input of the student to the system.
The ‘output’ object has a ‘feedback display’ which communicates the feedback of the system to the user and an ‘assignment display’ which shows the assignment on screen.
The ‘input’ object has an ‘input student’ which takes care of the input of the student and an
‘experiment input’ which communicates the settings of the experiment. The input of the student to the system consists of the predictions of the student and possible help questions from the student. The settings of the experiment consists of the relevant objects and the values of the variables. The responsibilities of the ‘interface’ object are:

- display the feedback and the assignment
- take care of the input from the student to the system
- take care of the input of the settings of the experiment to the system
CHAPTER 4. ARCHITECTURAL FRAMEWORK

Figure 4.7: Assignment object

Figure 4.8: Interface object
Chapter 5

Conclusion

In this thesis I have described an educational design for the seesaw problem and a software design for a system implementing this educational design. The purpose of the educational and software design was to build a system whereby I could analyze the question: by making the domain more concrete, would it be possible to teach children about domains that are normally at a too high cognitive level for them?

The resulting architecture described in Chapter 4 can be used as a basis for the software design. The resulting software design consists of a framework which can be used for different domains and problems. The advantage of such a framework is that you gain time and effort when creating new educational software for other problems and domains. The educational design and framework has already been made, so for a new problem just the specific software design and implementation has to be made.

Each object of the framework can be filled in with the specific needs of the different problems. The resulting software architecture can be made as ‘intelligent’ as needed. The framework has relatively independent spaces where intelligence can be used.

This framework is suitable for problems that are similar to the seesaw problem. The characteristics of these systems are:

- a simulated device
- the possibility to experiment with this device
- answering questions about this device, in particular outcomes of experiments
- a clear physical model, physical laws and parameters to reason about experiments

To answer the research question: “if the domain of physical equilibrium is presented at a more concrete way, will children at the higher classes of elementary school understand the concepts of the domain?” an educational software environment had to be build. But before such an environment could be built two other questions had to be answered. First what kind of educational environment to use? And second, how the abstract laws of physical equilibrium can presented in an concrete way.
The first question has been solved by the decision to use a discovery environment, so that the child is able to experiment on his own with the domain and this way remember the concepts of the domain better.

The second question of how to present the abstract laws of physical equilibrium in a concrete way has been solved by the following two decisions. Firstly to use a seesaw as a concrete example of an equilibrium device and secondly to split the material into different levels so that student would gradually be exposed to the material.

However the question whether children will understand the concepts of the equilibrium domain when these concepts are presented at a more concrete way, has not yet been answered because the actual specific design and implementation would exceed the scope of this thesis. But with the framework it should be possible to implement a design which could test this question.

A point of criticism with regard to the resulting framework is that the framework is only suitable for a very restricted set of domains and problems. An interesting next question would be if it is possible to build a framework that is suitable for more domains. When such frameworks are developed it would also be easier and less work to create educational systems. It would be nice if some framework could be designed where teachers can very easily make their own educational systems. Such an attempt has been made with the Simquest environment (Joolingen et al., 1997), but this environment is even more restricted than the framework of this thesis. An advantage though is that the environment is easy to use for teachers. Such systems could be the future, so that teachers can more easily use educational systems for their teaching program.

Another point of criticism is that there is a general description of where some intelligence can be implemented into the framework, but the actual techniques have not been described. In a next study it could be investigated which techniques are effective to implement intelligence. Especially when the framework should be kept as general and simple to implement as possible, this would be a difficult question.

So in conclusion, the framework of this thesis can be an interesting starting point to conduct more research to general frameworks, so that educational systems can be constructed more easily. This would be useful because intelligent educational systems have some advantages over human teachers and could be used very effectively besides and as replacement of human teachers. Especially this would be useful in the future when more individual teaching will be needed.
Appendix A

Objects

Below the objects are presented that are secondary helper objects of the main objects described in Chapter 4. For these helper objects only their responsibilities are described.

Assignment display
Responsibilities:
- show the assignment on screen

Feedback display
- show the feedback of the system to the student on screen

Figure A.1: The prediction and outcome comparer object
APPENDIX A. OBJECTS

Prediction and outcome comparer
Responsibilities:

- knows prediction student
- knows outcome experiment
- determines if prediction was right

![Diagram of Outcome determiner object](image)

Figure A.2: The outcome determiner object

Outcome determiner
Responsibilities:

- knows the laws that should be used
- knows the settings of the experiment
- calculates the outcome of the experiment

Input student
Responsibilities:

- take care of the input of the student, such as the predictions and the possible help questions from the student
Experiment input
Responsibilities:

- communicates the settings of the experiment, the relevant objects and the values of the variables to the system

**Figure A.3: The assistant tutor object**

**Assistant tutor**
Responsibilities:

- monitors if the student makes errors during experimentation
- explain errors to the student

**Explanation domain**
Responsibilities:

- explains errors to the student

**Assignment creator**
Responsibilities:

- construct an assignment
- retrieve an assignment
**Assignment constructor**

Responsibilities:
- complete problem template

**Assignment retriever**

Responsibilities:
- looks for an loads an assignment from file
- Takes care that the student will not do an assignment twice

**Formal domain description**

Responsibilities:
- knows the domain laws
- knows the domain objects
- knows explanations domain

**Laws**

Responsibilities:
- knows the laws that play a role
Law
Responsibilities:
- knows the variables that play a role
- knows the relation between the variables
- knows the level(s) to which the law applies

Domain objects
Responsibilities:
- knows the objects that play a role in the domain, for example a seesaw and blocks

Domain variables
Responsibilities:
- stores the variables that play a role in the domain, for example distance to the fulcrum and mass

Domain pictures
Responsibilities:
- Stores the pictures that play a role in the domain, for example the pictures of the 'domain objects'
Cognitive tools
Responsibilities:
  • storing the different cognitive tools, for example a calculator or a ruler

Assignment template
Responsibilities:
  • knows the basic structure of the assignment, the template where the variables and objects can be filled in

Knowledge of the student
Responsibilities:
  • stores the knowledge of the student

Learning style student
Responsibilities:
  • determines learning style student
  • knows learning style student

Diagnose mistake
Responsibilities:
  • diagnose the cause of the mistake
Figure A.7: The formal domain description object

Figure A.8: The laws object
APPENDIX A. OBJECTS

Figure A.9: The domain objects object

Figure A.10: The cognitive tools object
Figure A.11: The learning style of the student object

Figure A.12: The diagnose mistake object


