Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik
2004-08-18

Usability grading of a learner adaptive system

Abstrakt

The need for individualised support is important as sizes of classes increase. We present a study of a learner-adaptive system – a system that adapts to improve the learning experience of the user – for spelling in primary school. Together with teachers we have designed and evaluated a specific technique *evolution of educational content* (first explored by Sklar and Pollack, 2000) that refines spelling exercises based upon the student's earlier success. The method of triangulation is used to combine qualitative with quantitative measures for determining the usability of the system and *evolution of educational content*. The usability test derives from Squires and Preece (1999) set of *learning with software* heuristics – a framework that recognizes the importance of not only design but also learning outcomes. We found *evolution of educational content* highly usable for spelling training in primary school and as a complement to the conventionally taught curricula. The results showed an increase in spelling ability and demonstrated individualised learning with non-competitive interaction among students. Notably, individualisation occurred in spite of the absence of a student model.

Nyckelord: usability, learner-centred software, adaptive system, spelling, education

Författare: Marie Bodén
Handledare: Rikard Lindgren
Magisteruppsats, 20 poäng

Postadress	*Besöksadress*	*Telefon*	*Telefax*
Handelshögskolan vid Göteborgs universitet 47 54 Institutionen för informatik Box 620 SE 405 30 GÖTEBORG	Viktoriagatan 13	031- 773 10 00	031 - 773

Acknowledgment

Thanks to Milton State School, Brisbane, and especially the two teachers with students that were involved in the design and testing of The Magic Spell. Your help, advice and positivism have been a great ingredient for this work.

Thanks to Mikael Bodén for help with programming, generation of graphs and proof reading. To acknowledge the technical support I have received from Mikael Bodén, he is formally a second author of the paper *Evolving spelling exercises to suit individual student needs.*

ACKNOWLEDGMENT 2

1. INTRODUCTION 4

PROBLEM SPECIFICATION 5
A scenario 5

2. THEORY 7

LEARNER-CENTRED DESIGN 7
USABILITY 8
LEARNER-ADAPTIVE SOFTWARE 11
EVOLUTION OF EDUCATIONAL CONTENT 12

3. METHODOLOGY 15

RESEARCH APPROACH 15
METHOD FOR VALIDATION AND INTERPRETATION 16
EMPIRICAL STUDY 16
RESEARCH ENVIRONMENT: THE CLASSROOM 16
DATA COLLECTION 17
Statistics 17
Interviews 17
EVALUATION 18
Evaluation heuristics 18
A set of learning with software heuristics 18

4. THE MAGIC SPELL 21

THE SPELLING TASK 21
THE SPELLING ENGINE: MECHANISMS FOR GENERATING SPELLING WORDS 21
The spelling space 21
Word selection 22
THE INTERFACE 23

5. RESULTS 27

PROFILING DATA 27
Navigational fidelity 27
Match between designer and learner models 29
Appropriate levels of learner control 31
Prevention of peripheral cognitive errors 34
Personally significant approaches to learning 34
Recognition and diagnosis, recovery from errors 34
Match curriculum and teacher's customization 35

6. DISCUSSION 36

7. CONCLUSION 40

EVALUATION AND FUTURE WORK 41

8. REFERENCES 42

Appendix I 45
Appendix II 46
Appendix III 47
Appendix IV 49

1. Introduction

If we can find the optimal way of learning for every individual student, would that not be wonderful? Assume a learner gets an individuated curriculum, continuous feedback, and tailored assignments and exercises that challenge her thinking. Moreover, let us assume that the learner gets stimulation and encouragement by her peers and tutor, weighing the advances made at her own level. The aforementioned support enables a *constructivist* approach to learning. It may be hypothesized that the use of constructivist inspired educational systems increases the efficacy of learning. However, a more subtle, often overlooked but, with the introduction of information technology, increasingly pertinent aspect is their *usability* – how useful are such systems for the parties involved?

The psychologist Jean Piaget described conceptual learning as a two-pronged, active and constructive process, namely through *assimilation* and *accommodation*. Assimilation occurs as experiences are collated into new knowledge and is sometimes characterised as a formative process. The resulting imbalance (new knowledge versus old knowledge) triggers accommodation, a self-centred, refinement process by which the student tests various ways of integrating and re-organizing the old with the new knowledge – possibly in interaction with her peers. Since development is tightly interconnected with learning, Piaget was a strong advocate for supporting the student to control the direction and pace of her own learning process. When the students find something stimulating and interesting, they should be encouraged to investigate and learn. Piaget's theories of learning and development are commonly labelled constructivist and accord with the introductory scenario.

One accepted institute for learning is and has been for a very long time, school. However, it is often difficult for a single teacher to perform individuated learning in big groups of children, often 20-30 students per teacher. With the introduction of information technology there has been both scientific and commercial excitement surrounding the possibilities of using such technology for educational and pedagogical purposes. One type of educational computer software is known as *learner-centred* educational software, i.e. software that is adaptive to the learner's needs, pace and interest. A specific form of software developed according to a learner-centred design – here called *learner-adaptive* software – adapts its control mechanisms to fully encompass the specific learning needs of the user.

This thesis evaluates the efficacy, usefulness and usability of a particular learner-adaptive technique, here referred to as *evolution of educational content*, for computer-based learning. Evolution of educational content was originally proposed by Elizabeth Sklar (2000) in her PhD thesis. By careful incorporation into a computer environment the technique can fulfil a subset of the learning features identified above and thus provide partial support for constructivist learning (Sklar and Pollack, 2000). In collaboration with pedagogical peers, we incorporate Sklar's technique in a modified learning domain into a software environment and test the technique for real learning outcomes. As the potential scope of such an investigation is wide, our study focuses on the *usability* of this particular software. The software's general character allows us to carefully discuss the usability of such systems in general.

At the outset it is important to note that there is a well-developed theory surrounding usability testing of *user-centred* software e.g. Nielsen's (1994) usability heuristics, but conscious of the differences between learner-centred and user-centred software, we instead propose to rely on a list of informal rules-of-thumb for evaluating the usefulness of our learner-centred software, identified and discussed by Squires and Preece (1999).

Problem specification

A scenario

In a typical grade four classroom, the teacher will work with a new spelling rule in weekly cycles, exemplified using a set of words, every week. The students work in specific spelling workbooks and are tested on the week's spelling words. On occasion the teacher tests previously tested spelling words to ensure that students entertain their spelling abilities.

There is an incredible spread in reading, writing and spelling abilities among the students within a single classroom. Each and every student needs support at their own level and for their own specific difficulties. In a classroom with 30 students it is difficult for a single teacher to find time to offer personalised support. Sklar and Pollack's constructivist learner-adaptive software thus potentially offers useful support and relief for the teacher.

We set out to verify Sklar and Pollack's three advantages. Individualised learning is desirable in domains for which the student's have a varying expertise. Does *evolution of education content* support individualised learning for spelling in a classroom scenario?

It is important to run tests to find out how well this technique will function in a classroom situation. The users play an important role in securing the best outcome of the performance of the technique. Here we identify two groups of users, teachers and students.

We are concerned with porting the technique of *evolution of educational content* to the domain of spelling – The Magic Spell is developed. According to Sklar and Pollack such ports do not imply a severe domain-dependent development effort. Their technique would thus work well with spelling tasks.
Moreover, we wish to verify the learning outcomes observed in Sklar and Pollack's study. Compared with typing, spelling requires more understanding on the student's behalf for achieving success.

Finally, this study investigates to what extent, if any - can this technique be a support for the teachers in their profession? We focus on the process of learning and how the technique can be a support for teachers. Is evolution of educational content a good compliment to the traditional teaching?

For a systematic evaluation we rely on usability testing in accordance with Squires and Preece’s heuristics for testing usability.

2. Theory

Below follows a presentation of former research in areas connected to our study.

Learner-centred design

Learner-centred design (LCD) is an offspring from the area of user-centred design (UCD). The basic incentive of user-centred design is to support the design of a system that is easy to learn and use for all of its future users. To ensure appropriate design support, the designer identifies and maps all user needs, backgrounds and preferences in relation to the purported software, and involves the user in the process of software design and development. By involving the user throughout all the phases of the design process, she is able to give instantaneous feedback. User-centred design presents the underlying functionality in a user-individuated way. As an example, a spreadsheet program offers the user a wide repertoire of functionality. User-centred design attempts to present the repertoire of functionality so to maximize *accessibility* for each user.

The need for learner-centred design arose when designers of educational software found the principles from user-centred design undermine or misconstrue principles for design of learning support (Norman and Spohrer, 1996). Applying user-centred design on educational content, results in software that presents a structured analysis of the curricula (Norman and Spohrer, 1996). A structured *conceptual* view of educational content is useful, but in learner-centred design the focal point is on the student and her learning triggers – the user should engage in the underlying meaning of the content (Mayes and Fowler, 1999). As a result, a system, designed with the learner in mind, should *direct* the student to the appropriate functionality or activity embedded in or controlled by the software. According to learner-centred design, the software should adjust not only the presentation of the curricula to its learner but more importantly adjust or adapt the appropriate functionality and activities.

Mayes and Fowler (1999) argue that to achieve learning outcomes from software we must not focus on learning itself, learning is necessarily a by-product of understanding on the learner's behalf. The authors suggest that the design of educational software should focus on creating effective tasks which will support and benefit the student's learning. Student-system interaction is indicative of learning outcomes. The system can thus respond in ways to put the student in indirect control through its learning outcomes.

On a more general note, Norman and Spohrer (1996) point out that it is necessary to be aware of both user-centred and learner-centred aspects in good design. Without a well designed interface the learner would have to focus on how to interact with the interface. In addition, learner-centred design struggles with the differences between the learner's learning style and teacher's teaching style. The design needs to encourage the learners and concurrently satisfy the teacher's intentions of learning (Hsi and Soloway, 1998). We are thus not in a position to be completely satisfied with the current design templates.

Hsi and Soloway (1998) describe the general goal of learner-centred design as supporting software that increases the efficacy of learning through assisting the student to learn. If software, designed by learner-centred principles, is being used as a complement to traditional classroom teaching, it is justified by means of its ability to help the individual learner improving the understanding of the subject matter.

Learner-centred software should also include elements that will increase the user's interest in their own learning, to promote further investigation and learning even after finishing working with the computer. More generally, the aim of the system does not only include promoting computer-based learning but also the learning that may result after system interaction (Hsi and Soloway, 1998).

Norman and Spohrer (1996) describe learner-centred design as a principle that recognizes its users' different needs at different stages in the learning and that each learner has their own learning style. Furthermore they argue that learner-centred designed software should recognize when teacher aid or other forms of support are needed. The system should enable the appropriate feedback for encouragement to continue. Noteworthy, one of the major advantages with computer-based learning is the possibility of continuously give the learners appropriate and encouraging feedback in a timely manner. Norman and Spohrer (1996) emphasize the importance of encouragement; it can be the trigger for an enthusiastic successful student.

In summary, we need to design the educational software so that it suits every individual learner and supports each learner's understanding. This is not a new thought but using the information technology medium to achieve this certainly is. As such technology develops, there is a growing set of tools that we can use.

Usability

Usability is a central concept in Human-Computer Interaction (Löwgren and Stolterman, 1998). The purpose of measuring the usability of a system is to try to make better and more useful computer systems.

> Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. (ISO 9241:11).

By measuring the usability of a piece of software we find out how the user experiences the program, how well they perform in their tasks, how hard it is to learn the program and how flexible the design is. The process of usability grading can consist of both qualitative and quantitative methods and it can be performed as a predictive, formative or summative evaluation. The usability grading is a method for assessing the quality of a system, program or technique.

Commonly, usability grading is performed along a checklist such as Nielsen's usability heuristics:

- "Visibility of system status. The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- Match between system and the real world. The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- User control and freedom. Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- Consistency and standards. Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- Error prevention. Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- Recognition rather than recall. Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- Flexibility and efficiency of use. Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- Aesthetic and minimalist design. Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- Help users recognize, diagnose, and recover from errors. Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- Help and documentation. Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large."
  (Nielsen, 1994, p. 30)

Nielsen's (1994) usability heuristics are used for predictive evaluation of software. The heuristics are easy to follow and it is relatively quick. Importantly, Squires and Preece (1999) contend that Nielsen's usability heuristics usually fail to consider the aspects of socio-constructivist view of learning, since there is a close interaction between usability and learning issues, and checklists do not encompass learning issues.

Squires and Preece (1999) suggest an alternative set of 'learning with software' heuristic principles as a means for integrating usability grading with learning issues. Squires and Preece's work adopts Nielsen's heuristics and identify salient learning issues that should figure in evaluations of educational software.

- "A need for a *match between designer and learner models* is implied by considering intrinsic feedback and the relationship between learner and designer model. […]
- A requirement for *navigational fidelity* is apparent when navigational structure, cosmetic authenticity, limited representation of the world and superficial complexity are considered. […]
- The need to consider *appropriate levels of learner control* follows from a consideration of learner control and shared responsibility, self directed learning, tailoring and consistent protocols. […]
- The need for the *prevention of peripheral cognitive errors* is implied by the relationship between complexity and error prevention. […]
- The requirement for *understandable and meaningful symbolic representation* follows from a consideration of representational forms and the use of symbols within and across applications. […]
- The need for *support personally significant approaches to learning* follows from a consideration of multiple representations, learners' support materials and meta-cognition. […]
- The need for *strategies for the cognitive error recognition, diagnosis and recovery cycle* is implicit from the discussion of pedagogical techniques.
- That there is a clear need for a *match with the curriculum* is evident from a consideration of curriculum relevance and teacher customization. […]" (Squires and Preece, 1999, pp. 479-480)

Squires and Preece also identify more support for their main contention that the standard heuristics are insufficient. Established usability heuristics do not cope with innovative software (Heller, 1991). Such heuristics do not allow for novel or alternative models of pedagogy and teaching strategies (Winship, 1988). Evaluating different subject areas may require different selection criteria (Komoski, 1987). It is difficult to indicate relative weighing for queries (Winship, 1988). When comparing similar educational software the checklists tend to focus on similarities rather than differences and heuristics fail to consider the teachers' uses of the software (Squires and McDougall, 1994). On a more cautionary note, Squires and Preece suggests that their guidelines should be seen as a starting point for an evaluative framework rather than a rigid set of rules.

Usability grading of educational software does not come without controversy. Mayes and Fowler (1999) describe the major problem with usability testing of educational software as a paradox. On the one hand usability is to prove the obvious use of software. However, considering that the goal of educational applications is to support understanding which leads to learning (deep learning), it is far from clear that 'obvious use' leads to the goal. Mayes and Fowler classify educational software into three different groups, primary, secondary and tertiary courseware. The three groups of courseware support different kinds of activities and therefore the usability of each group should focus on different aspects:

1. Primary courseware focuses on mediating an educational content.

2. Secondary courseware focuses on creating good learning environments by modelling situations or using programming environments.

3. Tertiary courseware is based on using former results from students and enabling the students to discuss the results.

At the tertiary level one of the major issues is to keep a dialogue with the student. The motivation behind the third level of software is that dialogue is the foundation in all education. Through questioning, the student develops an understanding (Laurillard, 1993). The tertiary courseware is a useful complement to conventional classroom teaching since teachers have limited time for each individual student and therefore less time to help each student with personal, reflective thinking. Mayes and Fowler suggest the tertiary courseware as part of a solution to this resource problem. Measuring usability in tertiary courseware is ultimately about evaluating the efficacy of learning. Efficacy is difficult (and contentious) to assess. However, Mayes and Fowler suggest measuring the simple frequency of use as a rough but indicative replacement to learning efficacy. If learners are attracted to the software based on dialogue, they will probably find the usage of it valuable in their studying.

Learner-adaptive software

Educational software can be built in various ways. However, a few variants can be discerned (Sklar and Pollack, 2000). One principled way is to build the software on basis of an individualized student model (Greer and McCalla, 1994). The computer stores information about the individual student in a data model (e.g. a database or similar) and uses it to predict how a student will handle posed problems and how she progresses in her tasks. The model can be used to infer various user-specific pieces of information, such as which level of challenge is most suited for the student. The system monitors the model continuously for the selection of activities. The model adapts to the student-system interactions and aims to continuously mirror student progress. Otherwise the system follows more or less pre-specified paths.

A variant of building learner-adaptive software relaxes the constraints imposed by maintaining and using a student model and is based on the constructivist philosophy. All students should be able to learn in their own way without having to follow already made up tracks and pre-made levels (Resnick, 1997; Papert, 1993). The system needs to be involved in the learning (by monitoring progress) and adapt as the student progresses so the right level of challenge can be given. The constructivist approach to building software is open-ended and has no in-principle limitation in what activities the user may experience.

Sklar and Pollack, (2000) present what they call “an evolutionary approach to selecting content for educational games in a web-based learning community”, a good example of learner-centered tertiary courseware. Their system – with a clear exploratory component – is also an example of a constructivist approach to learner-adaptive software.

Evolution of educational content

In Sklar (2000) and Sklar and Pollack (2000) an *evolutionary* approach is proposed as a viable alternative to pre-leveled methods for realizing learner-centered software.
By "evolutionary" is here meant the means by which the user is directed to various activities in the system. In general and more specifically in nature, evolution is construed as the adaptation (through selective pressure) of a gene pool, embodied by the current population of living organisms. The principle has been implemented in computer software for purposes of solution search and optimization of many types of natural and engineering problems and processes. It was popularised by John Holland (1975) as genetic algorithms and evolutionary computation.
For our purposes the term "evolutionary" is construed in a much more specific and limited form: the selection of activities in the next phase of an educational game is influenced by the relative success of each and every activity that the user completed in the previous phase. The selection of new activities is realised by "genetic" operations including *mutation*, believed to realise actual genetic transfer, from parent to offspring, in nature. Moreover, the genetic operators are stochastic and may thus produce novel outcomes to be tested in the environment in which they "live".

Sklar and Pollack (2000) implement their technique "evolution of educational content" in a particular setting, analyse some of the technical aspects and present some preliminary results from using the technique in a classroom environment. More specifically, their software implements evolution of "suitable" keyboarding exercises. The goal for the system is to adapt as the players learn to type and to provide suitable challenges for all players.

The keyboarding exercises operate over the internet so students can "play" against each other. In the game the student faces 10 words at a time. The student is instructed to type the words as fast as possible and while the student types through the list of words, the system keeps track of the actual time taken for each of them.

The system retrieves words from a large database consisting of approximately 35,000 words of varying typing difficulty. In pre-levelled systems, the words would be presented in a pre-defined order, perhaps in a pace – adjusted according to measures of success – suited to the student. A model-based approach could log errors made and new words could be chosen so as to iterate words or word types for which errors occurred. Now, as mentioned earlier the constructivist approach to learner-adaptive software goes one step further by relaxing the need for a model. In the "evolution of educational content," there is no model, but words and thus typing activities are selected on basis of genetic operators and may consequently take novel paths through the space of possible typing activities. The control of the program is thus highly influenced by the success and failures of the individual user.

Using a pre-specified program control, a pre-leveled typing system could operate on the words themselves (they would all be "tagged" with a level). Evolutionary approaches partly remove the need for program control, but add the complication that we need to

define a space of “codes.” The codes correspond to words in an indirect manner, just as a set of genes translates into an individual organism. In the field of evolutionary computation, codes are often numeric vectors and the genetic operators modify these vectors numerically. As an example, a mutation operation may change one or two values in a vector of, say, 10 values. The magnitude of the mutation has an effect of the relative similarity of the previous vector and its offspring. For this imposed similarity effect to translate into similarity at the level of words, activities or organisms, the code-space needs to “mirror” the space of words, activities or organisms. The code-space can be high-dimensional and as such it provides means for expressing different types of difficulty of the domain to be learned. Sklar and Pollack (2000) suggest that the keyboarding code-space is a seven-dimensional space. In their application, dimensions correspond to typing features, namely (1) word length, (2) keyboarding level (as defined by a particular standard), (3) scrabble score, (4) number of vowels, (5) number of consonants, (6) number of 2-consonant clusters, and (7) number of 3-consonant clusters.

Initially, 10 points in the code-space are selected on a more or less random basis. Noteworthy, of the 90 million possible vectors, the dictionary accounts for 6074. The code-space is thus very sparse. However, using a method which Sklar and Pollack (2000) call “reproduction through sampling” the system generates only points for which a word can be identified. The student is tested on the 10 words and typing speed is recorded. The words are ranked according to typing speed (analogous to evolutionary selection) and divided into two groups: 5 words which need further practice, and 5 words that were handled well. Now, genetic operators are employed. For the 5 words that need further practice, 5 new points are generated by small mutation (meaning the new words will be similar to the old ones) of the 5 original codes. For the 5 words which were deemed successful, the 5 original codes are subject to large mutation. The large mutation means that the computer will make a big step in the space to find words classified in another group than the correctly spelled words. The resulting 10 codes will thus *exploit* the space in which words were handled less well by iterating the same typing features, and concurrently *explore* new parts of the code-space, replacing those words which were handled well in the previous instant. This process of testing, selecting and mutating is repeated multiple times. Students will consequently meet challenges that are a result of their success and failures of the previous activities. Notable, there is no pre-specified path that the student embarks into. The paths are solely and dynamically determined from the typing results.

Sklar and Pollack tested the system on 44 fourth and fifth grade students in the USA. They showed that the system adapts according to the capabilities of the individual student, showing more of the typing domain to the student who is ready to see it. Moreover, they showed that the typing activities experienced over the course were more varied than with a standard pre-levelled curriculum. Finally, the learning effects (though not exclusively attributed to the use of their system) were significant. 85% of the 44 students increased their typing speed.

Sklar and Pollack (2000) acknowledge three prevalent advantages of using an evolutionary approach in educational games.

1. There is a possibility of individualized learning. A system that adapts in accordance with the individual student's performance and in real time can provide suitable challenges and encouragement for the student proceeding in their learning. Correlation studies between variables in their classroom test provide evidence to support that students learn with the evolutionary approach and that they learn in different ways.

2. Student experiences are not pre-determined. Their analysis shows that the students embark into very different levels and that they are challenged with words they normally would not experience using pre-levelled software.

3. There are potentially less costs for development of educational games. The "evolution of educational content" technique is not limited in domain. As long as a code-space can be defined, the technique extends to any domain. There is thus less effort involved in developing new educational scenarios. Most prominently, there is no specific need to define levels and paths through the curricula.

3. Methodology

In this chapter the research process and approach are presented and justified. A plan of qualitative and quantitative tests is provided. In addition, the initial design considerations for constructing the software to be tested for usability and learning issues are outlined.

Research approach

The study is intentionally constrained to evaluate the usability and learning issues surrounding one specific case – one system/configuration only – over a limited time. Hence, the context of this case needs to be addressed fully. A combination of qualitative and quantitative research methods is chosen. The approach of combining the two principal methods is known as *triangulation* and allows us to frame the problems and their solutions in a systematic way. Triangulation gives us a deeper and broader perspective of our study (Cavaye, 1996). Both quantitative and qualitative methodologies are generally concerned with the quality of collated data in terms of applicability, validity, reliability and accuracy (Patel and Tibelius, 1987).

We use qualitative tools – *descriptive* and *interpretative* – to elucidate, map and characterize the potentially subjective opinions regarding the usefulness and applicability of the learner adaptive educational software. A description of former studies from other researchers is made and the knowledge has been used for comparing and explanation of our found results. In a descriptive study such as this one, the investigation is usually limited to some aspects of the phenomena that you are interested in (Davidsson and Patel, 1991). The interpretive approach is a way to help the researcher understand human thoughts and actions in a social and organisational context (Klein and Myers, 1999).

Quantitative measures are used to objectively assess many operational effects observed through the use of the learner adaptive software. These effects include simple usage frequencies but also potentially causal relations between types of inputs and learning outcomes.

The starting point for the study is the article written by Sklar and Pollack (2000). For finding further literature in the research area, we focus on learner-centred software, learning with software and usability. Also, we study literature cited by the aforementioned article. Importantly, we formulated questions we wanted to investigate from Sklar and Pollack's article.

The study is of inductive character rather than deductive (in the latter case you would rather try to verify a hypothesis; Backman, 1998). In our study there are two user groups, the teachers and the students. The two groups give us two different perspectives and help us find insights into how well the system works in a classroom environment.

Method for validation and interpretation

The user's experience of how well the system works complements the quantitative data. Triangulation is a method that originates from the physical sciences, where scientists use a signal measured at two or more known but different locations to determine the unknown location of the source. In a qualitative study, the triangulation is a combination of several research methodologies in a study of the same phenomenon. Denzin and Lincoln (1994) describe four different basic types of triangulation. The first one, *data* triangulation, compares different types of sources of knowledge. A second type is *investigator* triangulation, involving more than one researcher (or observers) in the investigation, each framing the problem in their own way. Using multiple theoretical schemes when interpreting data is known as *theory* triangulation. *Methodological* triangulation involves using more than one method for studying the problem at hand.

We have chosen to work with a methodological triangulation in our study. Methodological triangulation allows us to select the most effective methods for each individual aspect of Squires and Preece "heuristics for learning" and later integrate the outputs. By using individual strengths of a variety of methods, their combination becomes even stronger (Easterby-Smith, Thorpe and Lowe, 1991). For example, it is like identifying a person by means of fingerprint, voice recognition and face recognition, all methods considered and accounted for. We intend to combine quantitative statistics, qualitative interviews and literature studies of relevant research.

Empirical study

The learner adaptive software was designed with both the technology and the users in mind. Users include both students and teachers, two groups with disparate motives and interests. One complication arises from the young age of the target students. We have thus chosen to work using an iterative development process involving the teachers' opinion and mainly focus on input and feedback from the teachers. In the initial stages, two individual teachers were interviewed separately. They were shown an early prototype of a spelling program and were asked to provide input and feedback on content and functionality before it was introduced into their classrooms. General ideas and comments expressed in the interviews resulted in the development of the fully functional software. Interview questions are found in Appendices II-IV.

Research environment: The classroom

Two classes of students from a primary state school in metropolitan Brisbane, Australia, were selected as subjects for this study. Class A consists of 12 grade 3 students and 8 grade 4 students (ages 8 and 9, respectively). Class B consists of 27 grade 4 students. Class A has a female teacher and class B has a male teacher, both teachers are very experienced, cooperative and interested in the project.

The number of both teachers and students is small which could introduce some degree of uncertainty in the interpretation of the evaluative results. Moreover, due to the small population of subjects it was deemed difficult to introduce reference groups. All students were subjected to the same tests and activities. Other school work could not be

interrupted and we can therefore not attribute outcomes exclusively to the use of the software. All these considerations should be kept in mind while evaluating the results.

Data collection

Statistics

Before the software was introduced into the classroom, all students took a written spelling test consisting of 15 words (deemed difficult enough for separation to occur). This was done to minimize the number of explanations to observed effects. Half the population received one test and the remaining half received another. After the completed study – when the students had used the software for a period of 5 weeks – each student was tested again, this time with the alternative spelling test. Consequently, all students were tested twice, before and after, with different words. By averaging a neutral trend can be observed since all 30 words occurred both before and after the test (but for different students).

Importantly, the learner adaptive software was designed to log all processed words with all associated information. The spelling performance for each individual student is thus mapped in detail and a profile is built. The data will be discussed in the next chapter.

Interviews

After discussion with the School Principal, the empirical study was set up in two classes with one teacher in each classroom. Both teachers have been included in the teacher evaluation. As an introduction, during the design phase of the Magic Spell both teachers were interviewed. This was done so we would get an idea of how spelling is currently taught and what principles the teachers find most important in spelling software. The teachers were also asked to give feedback of an early prototype of the Magic Spell. Since we did not want to disturb the natural learning environment we did not perform any interviews while the system was being used. After completing the test period both teachers were interviewed a second time. The interview questionnaire was based on the evaluation heuristics presented below. We kept the questions semi-open, as Kvale (1997) suggests, to ensure we were given the answers on the questions we wanted answered, but also to maintain an open mind for opinions, feelings and thoughts from the interviewed. All teacher interviews were recorded on a tape recorder.

The aim is to find out the usability of our system and we therefore needed to find a way of evaluating all user groups. The interviews with the teachers allowed us to confidently say that no student disliked working with the software. It was consequently decided to select a representative group of students for interviews. We randomly chose eight students and then picked the first five students that were available for interviewing. The five selected student logs were checked to ensure we had a representative selection of students, covering the whole range of performance. When the student interviews were designed we decided to only take handwritten notes to avoid any shyness a tape recorder may impose on the children.

Evaluation

The main purpose of educational software is to promote learning. As discussed previously (see Chapter 2), so far most evaluations of educational software have been based on checklists which do not consider and account for the interaction between usability and learning. The two aspects have thus been separated at the outset. Teachers who perform evaluations are often highly professional when learning issues are concerned. However, teachers have seldom been trained to observe usability factors. Conversely, usability experts are usually poorly trained to consider and understand learning issues – a phenomenon that occurs external to the system.
In many cases, this lack of mutual understanding has led to an unfortunate selection of computer software based on rather arbitrary and practical considerations, e.g. availability of evaluation copy, compatibility with the school's computer system, etc.

Evaluation heuristics

The set of 'learning with software heuristics' forms a foundation for performing a usability test of the Magic Spell. In this work we consider each suggested heuristics and consider how each can be tested. Early in our research we came to the conclusion that we needed to perform a mixture of quantitative and qualitative testing, triangulation. By adding a qualitative interview of both teachers and students, we ensure that the context and the users' experiences are considered in the evaluation. The purpose of interviewing the involved teachers is to get their opinion on how a technique like the *evolution of educational content* works in a realistic teaching situation. The quantitative testing aims to find statistical evidence of how well learning develops while the qualitative testing is seen as confirming or rejecting quantitative findings, elucidating user impressions and providing a basis for explanations to the quantitative data.

A set of learning with software heuristics

The use of Squires and Preece's list of heuristics (1999) rests on two main assumptions:

1. learning is considered from a (socio-)constructivist perspective – users actively and constructively form knowledge, and

2. the application of educational software is thoroughly based on the context – the main purpose and situation must be central for evaluation to be indicative.

On basis of Squires and Preece's suggested rules of thumb for usability evaluation we designed a set of questions for both teachers and students. Also we decided how each rule could be quantitatively tested.

1. As a first rule Squires and Preece put the importance of a "match between designer and learner models". The designer-learner match is based on how well the feedback is been working between the learner and the designer model. The student should always receive feedback on their performance. It is important that there are no conflicts between the students learning style and the design. The learning style and the design do not need to be in perfect harmony as long as they do not conflict.

We want to know if learning developed as an effect of how the program taught spelling. Also, is the design assisting learning and is the feedback appropriate for all students irrespective of level?
We ask the teachers how they believe that the students learn in the Magic Spell and how often they think it is necessary to use a piece of software like the Magic Spell to get the most out of it. The students are asked if they enjoy playing the game and why they think so.

2. The second rule of thumb is "navigational fidelity". It is important that the design does not mislead the learner so that she rather focuses on the interface than learning issues. Squires and Preece argue that good usability is attained if the interface is supportive and encouraging for all learning tasks.
   We focus here on finding out how the teachers and students experience the software. We question whether the user interface is guiding, enticing or tricking the student to learn, and whether the interface hides (possibly distracting) complexity? From the system log we find out about the frequency of usage.

3. The third rule is based on the concept in constructivism that learners should be able to guide themselves through learning, in their own individually chosen way. This is one of the philosophies behind hypertext and other web-based instructional systems. Squires and Preece call this rule of thumb "appropriate levels of learner control". The students should have a sense of ownership and control. This sense is attained by providing an environment that adapts to the learner's performance in real-time.
   The teachers are asked if they find that students enjoy working with the software and how sufficient and appropriate they find the feedback for their students. The students answer questions on how hard or easy they find the spelling and how much they enjoy playing "the game."
   Quantitatively we look at the impact the learner had on how the words are selected and how (and how well) the spelling space is explored.

4. The "prevention of peripheral cognitive errors" is the fourth rule in Squires and Preece's list. A learner makes mistakes while learning and often needs to make mistakes to learn. It is therefore important to distinguish between cognitive errors and annoying peripheral usability errors.
   We ask the students whether they find it hard to understand how to work in the Magic Spell and if they find it hard to understand what actions to consider when making a mistake or error. The teachers are asked if they notice any particular problems that their students have in their interaction with the system and if there is anything students find hard to understand. We also ask how they find the interface and how well they think it is working.
   Quantitatively we look for consistency of errors, e.g. if errors are reasonably consistent and if other errors than spelling errors occur.

   According to Squires and Preece in educational software there is a need for the student to feel familiar with the icons and symbols used in the interface. The

intention of a symbol should be obvious to the learner and in general the interface should require a low cognitive demand. This rule is tested together with rule number four. As will be shown, the two rules interact.

5. The fifth rule considers “personally significant approaches to learning” which means that no matter what learning style the learner has, she should be catered for in an educational system. An adaptive system could support different learning styles, for example by using different presentation styles. In our evaluation we ask teachers about the different learning styles present in their classes and how well they think the system caters for the different learning styles, or if there are students and learning styles the system does not cater for. The children are not asked about this rule, since they are unlikely aware of their own learning style. Quantitatively we look to see if there are different patterns in their spelling space.

6. The strategies of recognition, diagnosis and recovery from errors are an outcome dependent on the pedagogical techniques used in the system. A major belief in constructivism is that all students learn by their mistakes and therefore the educational systems should be based on strategies supporting fault and mistake recovery. Squires and Preece suggest that a constructivist learning system typically is a rich environment where the students can discover different solutions to their problems. It is hard to build a system with our goals which encourages the students to find different solutions; in spelling there is normally only one way a word is spelled. In our inquiry we try to find out how the teachers find the students managing the software when they make spelling mistakes and also how the students experience their own failures.

7. A match for the curriculum is evident since there is an educational plan in every year to be followed. A constructivist model should respond to each individual student’s needs emerging while learning. The demands of following a fixed curricula forces teachers to find different software for students with different learning styles (Squires and Preece, 1999). A good, constructivist software is a piece of software that complies with the teacher’s methods and gives the teacher a chance to tailor the software for special needs. We enable the teacher to influence the design and ask them to reflect how useful the software is in their teaching.

4. The Magic Spell

This chapter describes the technology employed and interface used by the spelling program used in the study. The central concepts are developed and exemplified. These are needed to appreciate the quantitative analysis presented later. A rough understanding of the interface is needed to contextualize the comments made by students and teachers.

The spelling task

Hannafin and Land (1997) argue that technology is best used as a complement to traditional learning (human teacher-student interaction) but as a complement it can be beneficial for the learning process of the student. The Magic Spell is thought to be used as a complement to teacher supervised learning. The program itself does not directly teach the student the different spelling rules needed to know in English. The Magic Spell is spell training software.

Words are selected with the simple principle proposed by Sklar and Pollack for typing: Words that are incorrectly processed by the user are replaced with words that are similar (with respect to spelling; hence, enforcing further training), words that are correctly processed are replaced with distinctly different words (exploring other spelling constructs). This section develops the ideas of how words are selected.

The spelling engine: mechanisms for generating spelling words

The spelling space

The spelling space is the code space from which words are selected. Each word maps to a point in this space. Reversely, for a point in the spelling space, a set of words can be identified within a radius.

The mutation operator, described by Sklar and Pollack and inspired by evolutionary principles, is spatial in the sense that a new point is stochastically selected on basis of distance from an original point. The space can either be discrete (as is the case for real genetic data) or continuous (as is the case for most dimensions in Sklar and Pollack's work). To encompass a wide variety of outcomes, discrete spaces are typically much larger than continuous. If good coverage (of a spelling session) is to be ensured, a small space is easier to control. Another problem noted by Sklar and Pollack is the existence of regions for which no words could be selected.

As many other languages, English has a diverse set of spelling rules and a substantial number of exceptions. The spelling space is constructed from the spell patterns, combinations of letter that can reflect various spelling rules. Also since it has been argued that word that rhymes are typically handled with similar ability (Treiman, 1997) we also included a number of word endings.

Examples defined by so-called regular expressions include

`.*c[eiy].*`	words that has the substring “ce”, “ci”, or “cy” (soft c sound).
`.*ie.*`	words that have the substring “ie” (as opposed to “ei”).
`^th.*`	words starting with “th”.
`.*[iov]?e?s$`	words that end with “ies”, “oes”, or “ves”.

The full list of patterns is provided in Appendix I.

Each word matches a number of spell patterns. As we make use of a large number of spell patterns and as some patterns never co-occur, we decided to use singular value decomposition (SVD). SVD is a technique that is used to compress a large amount of data into a more easily managed quantity. Landauer, Laham, Rehder and Schreiner, 1997, successfully used SVD in their project with essay marking as did Kintsch, Steinhart, Stahl and the LSA Research Group (2000) for summarizing tools.
In our project SVD ensures that the spelling space is tightly packed with words to avoid blind hits. The procedure is described in Bodén and Bodén (2004).

Word selection

We collected 3622 words from two web sources claiming to supply useful spelling words for grades 1-5 (according to the US primary school system; here labelled as level 1-5). Also, words that are frequently misspelled were included (here labelled as level 6).

As described above all words are represented using a vector in the spelling space. Unmodified, the selection principle proposed by Sklar and Pollack would choose between words of arbitrary difficulty. The user could easily be intimidated by the sudden appearance of words which are too complicated and we thus suggest using level as an additional parameter to adjust during user interaction. Specifically, the spelling space is searched as usual by means of a point subjected to mutation, but only the words belonging to the right level can actually be selected. The level can then be adjusted (in accordance with the individual user) to maintain a pre-determined constant performance ratio (average spelling faults per word).

For a first time user, the level is always set to one. The students will therefore always be tested on some of the most basic spelling rules at level one before entering higher levels. After the first set of spelling words the system determines what next set of words to test the student on is to be. The next set of words is determined on correct or incorrect spelling. If a word is incorrectly spelled, the system makes a small jump in the space and finds a word, regarded as similar spelling features to the misspelled word, to be tested. If the word is correctly spelled the system makes a large leap in the space, to find a new word – most likely representing a different spelling problem - to test the student on.

Sklar and Pollack rank the words according to the typing performance and replace the set with a new set reflecting the performance of all the words collectively. For spelling

performance is not usefully characterized by speed. Instead we rely only on the discrete information whether the word was spelled correctly or not. The new set of words is based on the individual words, correctly spelled words are replaced by words that explore distinct parts of the spelling space, and misspelled words are replaced by words similar words in spelling space thus exhibiting similar spelling features.

The interface

The student inserts a floppy disk (onto which all data is stored) and a CD ROM with all necessary general files. The program uses speech synthesis to communicate with the student. The program greets the student by his/her name and starts the main loop of the program.

(1) The program displays a list of six words (see Figure 1). The words are selected in accordance with the previously described algorithm. The user can click on a word to hear the program pronounce it. When the student is ready (he/she may take any time needed), “go ahead” is selected and the program enters the next step.
(2) All words are hidden and corresponding text fields are displayed (see Figure 2). The student jumps between the fields. At each field the program speaks the word. The student types in the word and may edit the spelling of it until the next step is entered by pressing “try spell”.
(3) The program checks the spelling of each word and disables all fields for which correct spelling was detected (see Figure 3). The misspelled words are shown with the correctly spelled word beside the text field. This step is repeated until all words are correctly spelled by the student. When all words are correctly spelled the program jumps back to step 1 and the whole procedure is repeated.

The student is not informed of which level he/she is at. Neither is the student informed of the total number of correctly spelled words. However, the student is rewarded with a “bean” (seen on the left hand side in Figure 1-Figure 3) whenever he/she manages to spell four (of six) words correctly during step 2. Moreover, the student is rewarded with a “magic wand” whenever six beans have been collected. Notably, the rewards are not based on level and total number of correctly spelled words. Instead each student is rewarded at his/her own level. It is only when 75% of all words attempted at the current level (when at least 10 rounds have been completed) that the students is promoted to the next level. The student is not notified of or rewarded by the upgrade.
To encourage interaction between students the rewarded wands can be “traded”. The rules of the program allow students with lower spelling accuracy but with persistence to gain as many rewards as those with high precision.

Figure 1: The user interface when the user steps through the word he/she is about to spell. Typing is disabled and the user can only move the speaking robot between words.

Figure 2: The user interface when the user is asked to spell each word in turn.

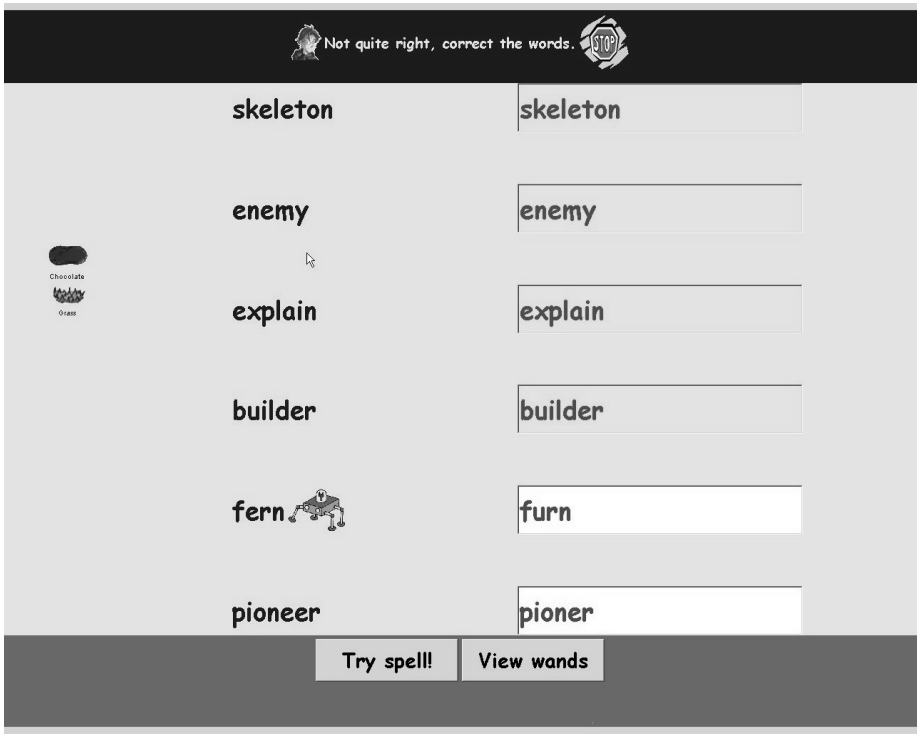

Figure 3: The user interface after the user has attempted to spell the words. Incorrectly spelled words are highlighted and the user is asked to correct their own input while provided with the correctly spelled word.

Teachers can monitor progress of the students using a separate program. The administrative program reads the log from the student's floppy disk and allows the teacher to change various settings. There are two main screens for monitoring the progress: spelling ratio on each of the levels the student has attempted (see Figure 4) and the spelling ratio for each of the spelling patterns (see Figure 5).

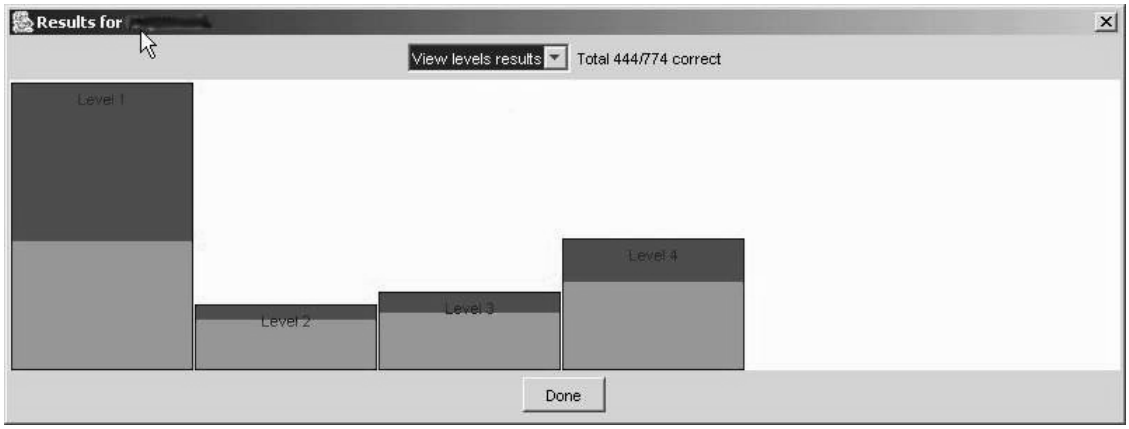

Figure 4: One graph presented by the teacher's program. A specified student's progress through levels can be monitored.

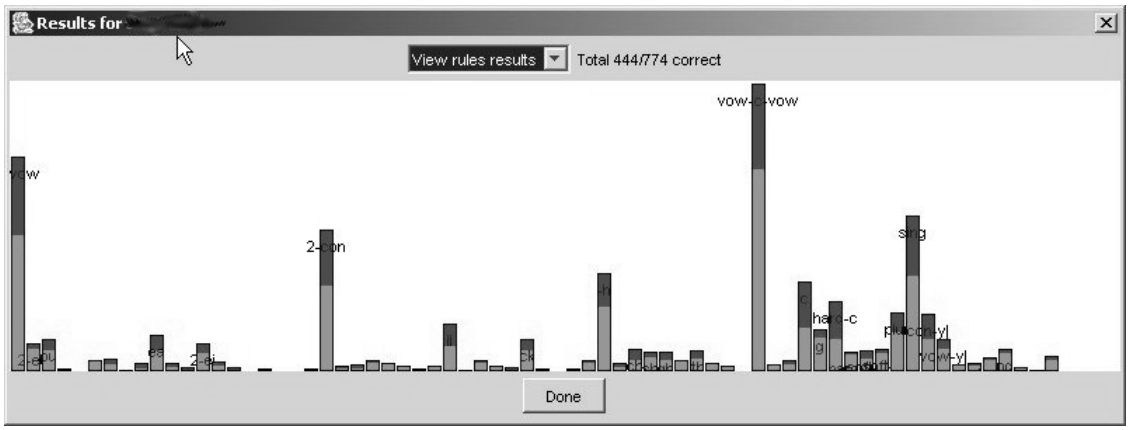

Figure 5: A second graph presented in the teacher's program, showing the spelling accuracy for all the spelling patterns used.

5. Results

Below follows a presentation of the results of the finished trial period. The presentation follows the set of learning with software heuristics, as discussed in the method chapter. The results are both qualitative and quantitative.
The qualitative analysis is based on interviews with school teachers and interviews with students.

Profiling data

The Magic Spell logs all words tested, which level the word belongs to, and whether the word was spelled correctly. Moreover, we can derive if this was the first time the word was presented or not. Finally, we can find out if the word was selected on basis of exploration (the previous word was correctly spelled) or exploitation (the previous word was incorrectly spelled).

The vocabulary consists of 3622 words and with spell patterns as presented in Chapter 3.

Navigational fidelity

Both teachers believe that the students improved their spelling ability by using the Magic Spell. Collecting as many wands as possible (the reward in the spelling game) appeared to be the primary goal for the students and encouraged them to work with their learning tasks. When the students were asked how they had enjoyed working with the Magic Spell, we received a united positive answer. The students thought the software was “a fun game” and the game made spelling more fun. The teachers consider the software as a complement to other spelling tasks they work with in the classroom.

In both classes there are children who have special learning difficulties such as hyperactive and English as second language (ESL). Generally the teachers found that the software works very well with most of their students, including the hyperactive students. The ESL students seem to have difficulties understanding the robot’s pronunciation and therefore needed extra attention from the class teachers while working on the computer. The ESL students do not necessarily hear all the English sounds and therefore have trouble spelling the words.

The students work in varying tempo and with varying persistence. After five weeks of usage, the students had completed a mean of 304 words (median is 282). The standard deviation is 197 words. With the help of the program, all attempted words are spelled sooner or later. If we look at the number of words correctly spelled in the first instance the distribution is similar. The mean is 194 words (the median is 187) and the standard deviation is 113 words over all students.
The variability in completion can not only be quantified in terms of number of words. The Magic Spell automatically adjusts the level when, after a pre-specified number of rounds (10), the student has shown consistency in accurate spelling. In Figure 6 the most difficult level (the final level reached in the five trial weeks) for the students is shown.

The distribution shows that most students (40/47) reached beyond level 1, of which some reached the 6th and top most level. Overall, the variability is high, the mean is 2.6 (with a median of 2) and the standard deviation is 1.3 levels.

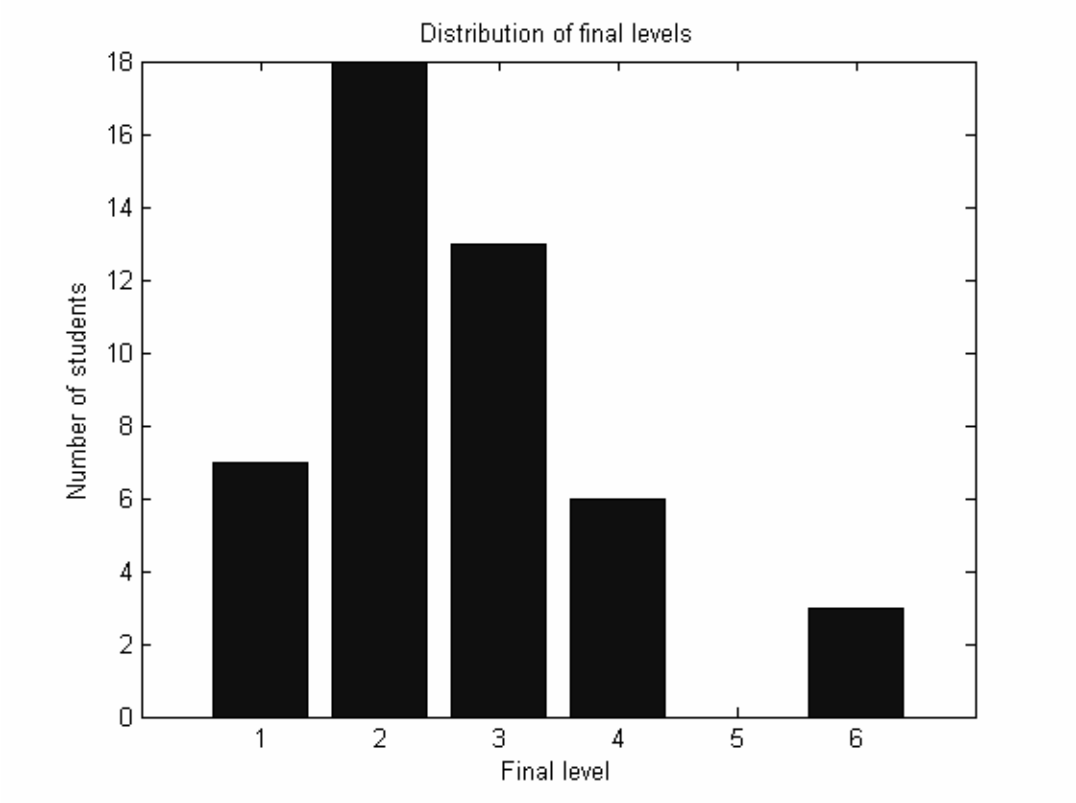

Figure 6: The distribution of the most difficult level attempted during the trial period.

If one instead focuses on the number of words that were completed in each of the levels, the distribution is slightly different. In Figure 7 it can be seen that the students completed almost 8000 words at level 1 and only a few at the higher levels. The mean level is 1.8 (the median is 1) and the standard deviation is 1.1 levels. However, this is mainly explained by the fact that all students start at level 1. Given that most students ventured beyond level 1, the tail would be longer with a longer trial period.

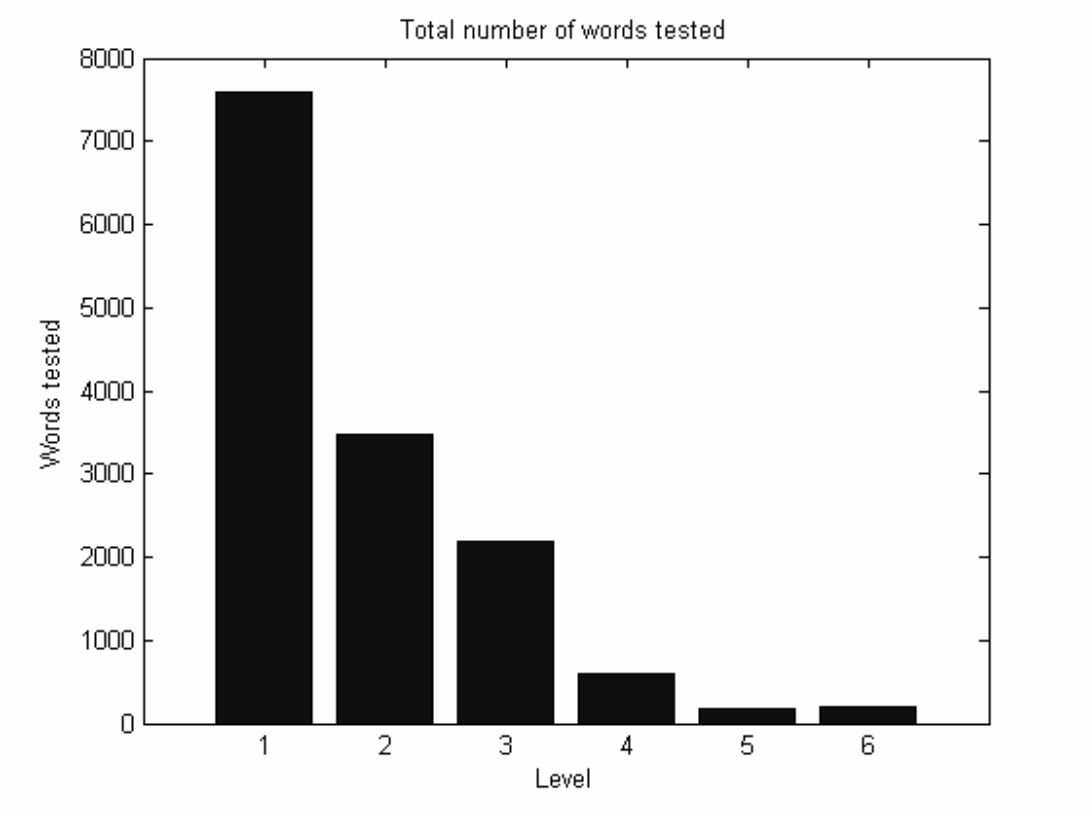

Figure 7: The distribution of words tested within levels (accumulated over the whole trial period).

Match between designer and learner models

When English is taught in the classroom the class teachers mainly work with the methods of rehearsing and repetition. The teachers encourage the children to read and then writing through the list of weekly words. Finally, they check their spelling. The procedure is repeated until the students know how to spell the words properly. Teachers believed that the program was consistent with the aforementioned procedure but the program worked with a much larger database of words. It was suggested by the teachers that appropriate time spent with the software would be 3 times a week, with 30 to 45 minutes each time for learning to be effective. All students thought they became better spellers after working with the system. The students did not appear to know how they actually improved their spelling in the Magic Spell.

To ensure a spelling system has a purpose in a classroom, we need to demonstrate the system actually improves the student's spelling. The Magic Spell continuously changes so it can provide a personalised challenge to each student. The system is based on the thoughts of constructivism but also considering the different difficulty levels on spelling words used at school. The system tests the students spelling ability at a level before moving up to a higher level.

By looking at the first ten rounds of words tested at level one, the results we discuss in Bodén and Bodén (2004) directly show us a significant increase in spelling. Specifically,

we need to find evidence that shows the student increases the probability of correct spelling when challenged with a novel word. Furthermore, if the novel words are reached by exploration a positive result is not based upon the student being presented with the same spell-pattern again. The probability of correct spelling when the student sees a word for the first time, not following a similar word, at the most basic level is shown in Figure 8. The increased spelling accuracy is obvious.

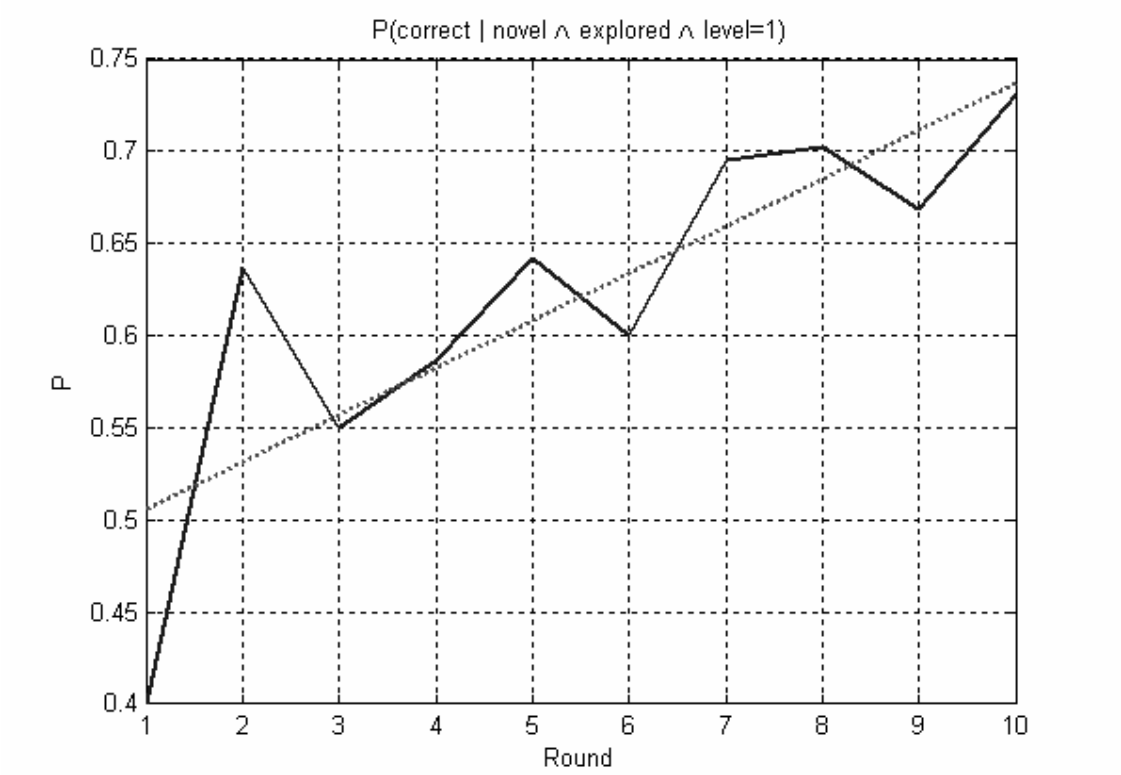

Figure 8: The probability of correctly spelling a word increases within level 1. The solid line shows the probability for each round (up to round 10) within level 1 over the 47 students. The dashed line shows the best linear fit to the data.

To see how the spelling accuracy continues to improve by using the Magic Spell we – as explained in Bodén and Bodén (2004) – also look at the probability of correct spelling in the last 25 rounds of the final level (the final level might differ between students since they have been working at different rate). The probability of correct spelling is still obvious as seen in Figure 9 but there is an increase in variations between students' performances, which can be explained by the small amount of students who finished 25 rounds in their last level.

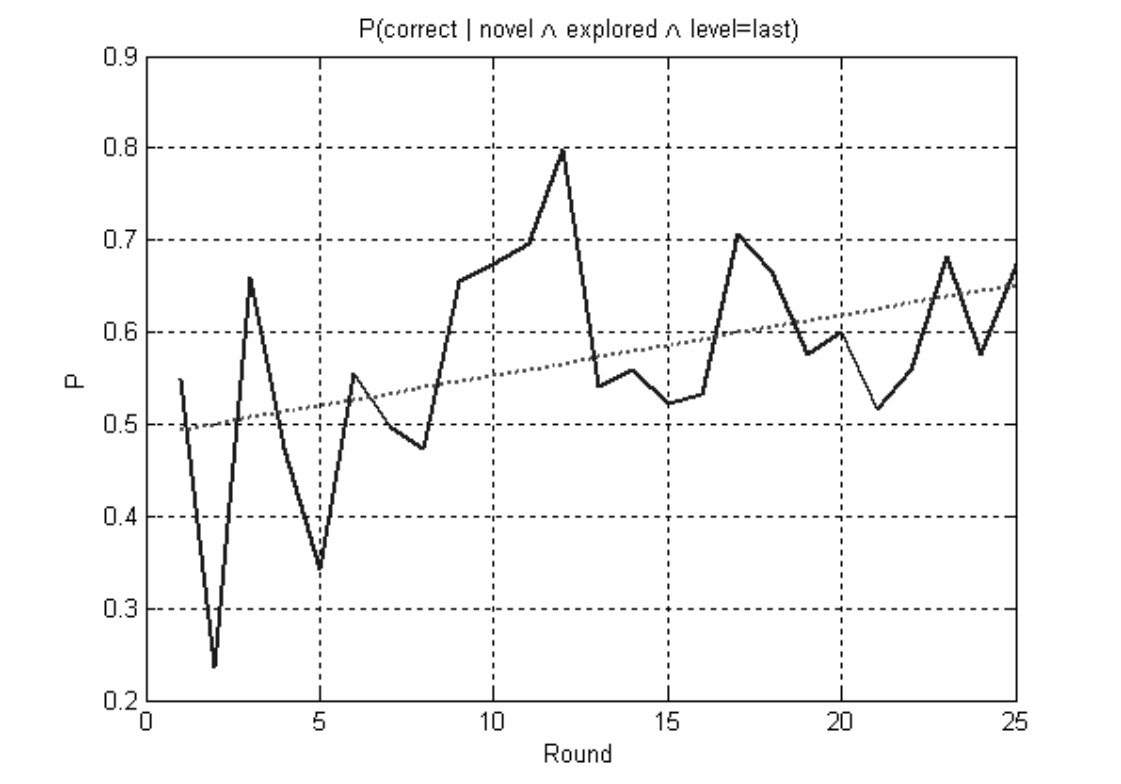

Figure 9: The probability of correctly spelling a word increases within the last level (for each individual student). The solid line shows the probability for each round (from 25 rounds prior to the completion of the trial, through to the final round) over the 47 students. The dashed line shows the best linear fit to the data.

The teacher prompted handwritten test of 15 words before usage of the Magic Spell and 15 words after the test period showed neither increase nor decrease in spelling. Over 45 students writing the test before and 44 after, the average correctly spelled words was 8 both before and after.

Appropriate levels of learner control

The feedback seemed to be sufficient for the students and the teachers found their students happy to work with the system. The students also agreed that the system was a joyful game and, even though they thought it got harder to spell the words after a couple of sessions with the software, they did not find it hard to win rewards.

According to the teachers the software also seemed to encourage the interaction between the students, on several occasions the teachers found their students helping each other and discussing spelling matters. The rewarding system with beans and wands also promoted the interaction between the students. Many students were keen to see other friends being successful in their work since that would mean that they were rewarded with a new wand that they might swap with. There was a noticeable difference between the younger, weaker spellers and the other students, where the young and weak speller would more

often get stuck on a word that they did not know how to spell. This very often happened when the words started to become more difficult to spell. The system clearly indicated when a student had misspelled a word. Although, when a child had problems with a certain spell pattern, one teacher suggested that it would be good if the system gives an indication on how to spell the word correctly. Even though this was a problem with the younger spellers, the same problem never occurred with the weaker spellers in grade four.

To find out the appropriate levels of learner control quantitatively, we look upon how well the spelling space is explored. The frequency of spell-patterns tested (shown in Figure 10) is relatively even when several students are plotted. However, the frequencies of individual students are different. The latter indicates that the system supports individual requirements (meaning students can take different paths in their work with spelling). This issue is further explored in Bodén and Bodén (2004).

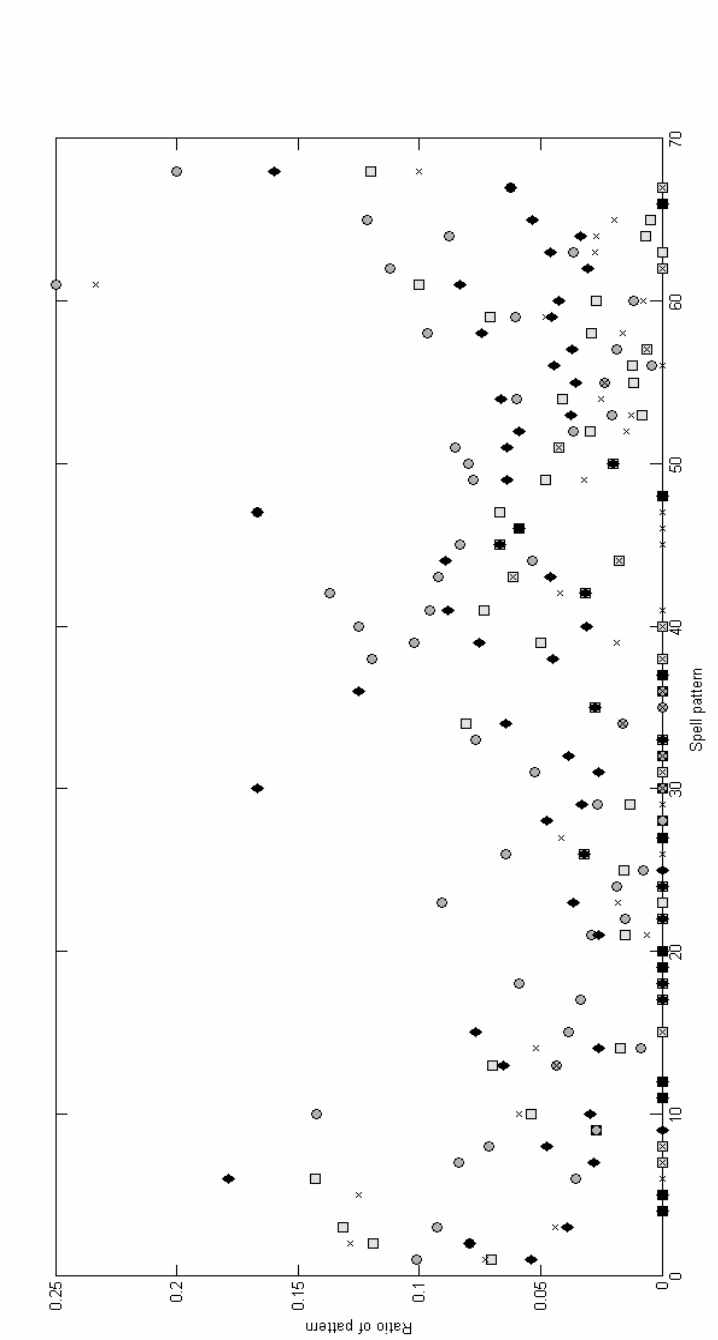

Figure 10: The coverage of spell patterns for four different students. The coverage of a pattern is normalized by its frequency in the word database.

Prevention of peripheral cognitive errors

Both teachers considered the user interface easy to work with for the students. Both teachers mentioned their fondness of a simple interface with bright colours and not too many distracting appearances. They said the students easily could focus on their spelling matters without distractions. The students said it was easy to know what to do and they almost never had any problems knowing which button to press. The feature of locking the system until a correct spelling was typed was considered good by the teachers because it worked very well with their ordinary spelling methods they use in the classroom.
The students needed a short introduction on how the system was working and after that they performed well without interruptions. Some students had problems understanding the robots pronunciation at the beginning and after two or three sessions with the software those problems were not apparent, according to the teachers and verified by the student interviews.

Personally significant approaches to learning

There are three different learning styles, auditory, visual and practical. Auditory learning means they learn by listening to the pronunciation. Visual learning is based on the appearance of words which is then remembered by heart. The third one called practical learning means that the student works with the different shapes of the letters by hand. There are examples of all three learning styles in both classes. One teacher thought it was very obvious which learning style the student worked with when they were at the computer. The students who used auditory learning style were the students who had a bit of a struggle understanding the robot pronouncing the words. The visual learning students were those who were most successful, they just read through the words and spelled them. The practical learning students were the ones who struggled most and the system failed supporting their learning style. The second teacher did not recognise the differences when observing the students by the computer but found the produced statistics more informative.

According to the teachers, the system is insufficient for ESL students as is, but with a better voice for the robot and maybe with some feedback the program would work for all students.

Recognition and diagnosis, recovery from errors

From what the teachers could discover it was clear to the students when they did something wrong. They did not have any frustration among the students about what to do in the system. The teachers' conception is in line with the students. They were satisfied with the feedback given from the system and the only problem recognised was when the students had moved up to a more advanced level. One teacher suggested it would be good to implement a part in the system that would educate the student on how to work with different spelling rules. The educational part could occur when the system recognises that the student has had problems with certain spell patterns. The probability of correct spelling when a word came up for the first time is 63 percent. Further, the spelling accuracy, when the children were tested a second time, is 66 percent. The variability

while estimating these probabilities is rather low (standard deviations over students are 0.07 and 0.08 respectively).

Match curriculum and teacher's customization

The Magic Spell received support from both teachers for fulfilling the requirements of use in a classroom context. The software worked well together with the teaching goals and there has been no problems integrating the system with the ordinary teaching schedule. There was a definite agreement on how useful the statistics, that the system produces, were for the teachers. The production of student results was a time consuming task they normally have to perform by hand. With the Magic Spell they could easily gain statistics every time the student had been working. Both adult testers said they would like to test the system again in the classroom but start working with the system earlier in the school year. Also, the involvement of the teachers during the design of the system gave the teachers a sense of positive impact.

The content in Magic Spell covered the traditional curricula in grade three and four well but at the end of the testing period some students faced some difficult spelling words. A suggestion from the teachers was that it would be good to let the students work for a longer period at each level and then gradually move up a level.

Other improvements would be a better voice on the robot, more hints/help if the student does not know how to spell a word. Another suggestion is to highlight the appropriate letters in a word so that the rule or rules become apparent to the student. Yet another suggestion was that it would be excellent if the student could see the graphs that show how their spelling has improved; the teachers thought could be encouraging for them. A last recommendation was that if it would be possible to show the student, if the word is supposed to be spelled with body, head and tails, which would definitely help for those children whose learning style is practical.

6. Discussion

This chapter discusses the results of the experiments, synthesizes interviews and statistics, and relates our findings to the literature and current research. The statistics we find in our results indicate how well the technique is working and the qualitative interviews show the users experience of working with the system (an application of triangulation).

Needs

The importance of a system that covers the needs of both students and teachers is important in an educational system (Hsi and Soloway, 1998). In the present work we have presented an alternative, bottom-up method for teaching and training students how to spell. The method complements the traditional initiatives in the taught curriculum, but exhibits added-value by being student-centred.

The developed and evaluated software is incorporating the simple principle of evolving educational content, but is also the product of a collaborative effort between the involved teachers and the designer. It is therefore not entirely surprising to learn that the teachers found the software to function flawlessly. We believe that the successful outcome indicates the importance of involving the pedagogically informed parties early in the software development process. The teachers observed that students approached the Magic Spell with big enthusiasm and with a high degree of effort.

Support

Norman and Spohrer (1996) describe a good learner-centred design as one that recognises when and what support is needed for the students. Both school teachers consider the software only as a complement to other spelling tasks and teaching – and the software was purposefully designed in accordance with this philosophy. The statistics the software maintains and presents, shows exactly how each student has performed and serves as a teacher guide for what abilities and problems each student has. The guidance allows teachers to commit human support to where it is best needed and suggests features that need further practise. In particular one teacher found the results report module extremely valuable. The students will by their performance ensure the system produces individualised spelling tasks and the teacher can also prepare specialised tutoring for each student.

Flexibility

Students should be able to learn in their own way without having to follow already made up tracks and pre-made levels (Papert, 1993; Resnick, 1997; cf. Bunt and Conati, 2003). The system needs to be involved in, and support the learning by monitoring progress and adapt as the student progresses so the right level of challenge can be given (Bull et al., 2003; Motschnig-Pitrik and Holzinger, 2002). A constructivist approach to building software is open-ended and has no in-principle limitation in what activities the user may experience. The teachers found that the technique individualised spelling exercises effectively. They thought further that the spelling problems were posed in an encouraging fashion – all students seemed to be content and active.

Peripheral outcomes
A system that encourages further learning outside the system interaction (Hsi and Soloway, 1998) has reached one of the major aims in educational software. In interviews with students they told us how they recognised their improved spelling when they did project work which required writing and while worked on related subjects.
Some students were encouraged when they were shown the statistics generated by the system.
Mayes and Fowler (1999) argue that "the learner needs to move effortlessly to the conceptual level, but then must engage with the underlying meaning. [ ] the software must make the learner think". The children were observed to have a playful attitude towards using the software, as also indicated by interviews with the students. However, when asked if they had noticed any differences in their spelling abilities the main part of the children recollected improved spelling when working with other tasks e.g. projects, writing stories and reports.

Collaboration, not competition
The teachers reported how the interactions between students had been successful in the sense that students wanted to help their friends with their spelling in a non competitive way. The standard deviations for figures that show the probability of correct spelling for novel and non-novel words indicate that the system keeps the success rate rather constant and consequently reduces the element of competition and encourages collaboration. None of the students worked through the system in an exact way so they did not compete with each other, they were interested in each other's success and they were eager to make sure friends gained new awards to be traded later.

Accuracy and effects
In Sklar and Pollack's article they show a relative increase in typing speed, 85% of their students improved their typing performance. In our research we also found a positive trend of learning outcomes. Both studies show the possibilities for individualised learning and support the principle of evolving educational content. We also agree with Sklar and Pollack, that the technique could be useful for many different curricula. We choose spelling as curricula instead of typing, spelling is a curriculum that put different demands on the student compared to typing. Noteworthy, as for Sklar and Pollack's work, the increase in accuracy can not be solely attributed to an increase in spelling ability.

Unfortunately the low number and selection of words rendered the teacher prompted handwritten test inconclusive. In hindsight, a selection of words better matching the ability of each student would make the test more sensitive for the improvement noticed from the computer study. We also acknowledge the small sample size as rendering the teacher prompted handwritten test insignificant.

Improvements
We performed a usability test to find out if this technology can be useful in education. The test has given us results that show us the system has a high degree of usability. Both

teachers and students expressed satisfaction working with the system , e.g. the students said the Magic Spell was an amusing way to learn spelling and the teachers received plenty of information about every single students spelling performance. The results also show a number of weaknesses that needs to be improved for a better performance. The system does not cover all learning styles (does not cover the practical style) but we did not find any proof that the system failed these students.

The teachers work through the spelling grammar after a pre-planned sequence while the Magic Spell follows each individual student's ability. Even though the teachers and the software work from two different pedagogical perspectives, the teachers were very pleased with the software. There was only one impact they would like to have on the system to make it function together with the ordinary spelling tasks. They suggested adding a control to ensure that all students are tested (and trained) on their weekly spelling words.

Learner-adaptive software
Working with learner-adaptive software allows all students to work completely individualised. The student indirectly decides what they are ready to learn at the next step. Papert (1993) and Resnick (1997) discussed advantages with such systems e.g. the possibility of supporting individualised learning styles and letting the single student explore and learn after their own ability. We believe the principles presented in our work represent one step towards better support for individualised learning in the classroom. According to the teachers, the Magic Spell is also well suited for students with concentration difficulties. The system maintained a reasonable level of challenge which partially explains why students with difficulties seemed to cope with the exercises as long as their fellow students.

Quality control
The quality control is increased with the Magic Spell, since the teachers can easily gain statistics on every student in their class. The results can then be used when planning individual tasks and for showing performance over a longer period.

Need for a student model
In our project, the Magic Spell, we found that the system individualised its tasks, that the students improved their spelling, and that the students explored the system in a non-predetermined way. These above mentioned findings were observed in the absence of an implemented student model. We believe it would be interesting to perform more research in this area to confirm weather a student model might not be necessary for a learner-adaptive system. To the contrary, a pre-specified student model may constrain the problems faced while experimenting with the software – partially denying the user a truly constructivist experience.

Importance of teacher involvement
Another observation we made when performing our study is the importance of involving the teachers early in the development of the software. Some of the results might be an outcome of the positive co-operation between the designers and the teachers. As for a

start, teachers found it difficult to understand how learner-adaptive could be a support in their teaching but after we showed them a first prototype it became more obvious to them and they soon became a good source of ideas pedagogically ideas for the designers. We found the combination of co-operation between the pedagogically experts and technical experts resulted in a product that used the technology and functioned well in a classroom setting.

7. Conclusion

In this section we conclude our findings and we suggest some research projects that we believe will be a natural continuation of our work.

A learner-adaptive system supports learners with an individualised learning environment. Our evaluation of the Magic Spell shows a positive trend in spelling accuracy. The Magic Spell uses the same principle as Sklar and Pollack (2000) use in a keyboard typing system. Sklar and Pollack observed an improvement equivalent to 85 % in their keyboarding test. Sklar and Pollack (2000) suggests that evolutionary technique is a cost effective technique for educational software and that the technique is easy to adapt to any curricula wanted.

A learner-adaptive system encourages a non-competitive environment for students. The students are not aware of how they work within different levels of spelling while the system maintains a steady level of difficulty to encourage the student. The non-competitive environment encourages a positive communication between the students.

Coverage of the possible spelling exercises varies with student – a sign of adaptation to individual difficulties. The principle of evolving education content transfers well into other domains with little system development involved. In the present work (and elaborated in Bodén and Bodén, (2004) singular value decomposition is introduced as a means to construct a space in which activities are organised according to a possibly large and sparsely populated feature space. Exercises are effectively selected by an evolutionary, stochastic process yet faithful to the previous successes and failures of the student. The overall development required for transferring the principle to other educational domains is basically confined to interface programming – little effort was required to get the evolutionary control mechanism to work in the spelling domain.

The following list identifies the main outcomes of our study that support the use of an evolutionary approach for learning.

- Learner-adaptive software is highly useable in education as a complement to the teacher.
- All students were worked happily and challenged with spelling.
- The degree of individualised spelling tasks were increased in the classroom.
- Good quality control for teachers.
- The system attracts all students including students with concentration difficulties.
- A positive learning effect is shown when using the system.
- The system encourages a non-competitive interaction between the students.
- A definite possibility for individualised learning.
- The Magic Spell support auditory and visual learning styles in spelling.
- No need for a student model in learner-adaptive systems

Evaluation and future work

There is still a lot of further research needed before we have found an optimal system to support learning in our schools. Our research has shown a positive trend that verifies Sklar and Pollack's work and we have also found more evidence that this technology might be worth investigating further. There is a need to run test periods with younger students to find out if the learner-adaptive software works as well with those students as with our students. Also running tests during a longer period for more validate results. What impact will learner-adaptive software have on teachers teaching in a more long term perspective? Both teachers made clear what steps they teach by and what spelling rules they were supposed to teach the students through each grade.

By using the Squires and Preece's suggested thumb rules for evaluating learning with software, we choose to work with a relatively untested method. Research and further analysis considering the thumb rules will be needed to confirm our findings.

We received useful feedback from both students and teachers on improvements for the Magic Spell. One suggestion was to include a section were the students can ask for hints on how to spell correctly. E.g. One teacher said it would be useful for the students to learn about certain spelling rules and that these would be acting as a guidance how to spell correctly. This will improve the strategies of recognition, diagnosis and recovery from errors which Squires and Preece points out in there thumb rules.

One reason for a positive outcome of the usability test is the involvement of teachers in the development process. In future work it would be valuable to confirm test results with teachers who have not been involved in development of the Magic Spell.

8. References

Backman, J. (1998). *Rapporter och uppsatse.* Lund: Studentlitteratur. In Swedish

Bodén, M. and Bodén, M. (2004). *Evolving spelling exercises to suit individual student needs*.URL http://www.itee.uq.edu.au/~mikael. (2004, May 21)

Bull, S., Greer, J., and McCalla, G. (2003). The caring personal agent. *International Journal of Artificial Intelligence in Education*, 13:21–34.

Bunt, A. and Conati, C. (2003). Probabilistic student modelling to improve exploratory behaviour. *User Modeling and User-Adapted Interaction*, 13(3):269–309.

Cavaye, A. (1996). Case Study Research: a multifaceted research approach for IS, *Information Systems Journal*, 6, pp 227-242.

Davidsson, B., Patel, R. (1991). *Forskningsmetodikens grunder*. Lund: Studentslitteratur. In Swedish

Denzin, N.K. and Lincoln, Y.S. (1994). "Introduction: Entering the Field of Qualitative Research", in N.K. Denzin and Y.S. Lincoln (editors) *Handbook of Qualitative Research*. Thousand Oaks: Sage.

Duda, R. O. and Hart, P. E. (1972). *Pattern Classification and Scene Analysis*. Wiley, New York.

Easterby-Smith, M., Thorpe, R. and Lowe, A. (1991). *Management Research: An Introduction.* London: Sage Publications, Ltd.

Greer, J. and McCalla, G., editors (1994). *Student models: The key to individualized educational systems*. Springer Verlag, New York.

Hannafin, M.J. and Land, S.M. (1997). The foundations and assumptions of technology-enhanced student centred learning environments. *Instructional Science 25*, pp. 167-202.

Heller, R. (1991). Evaluating software: A review of the options, *Computers and Education, 17* (4).

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Hsi, S. and Soloway, E. (1998). Learner-Centered Design. *SIGCHI Bulletin*, Vol. 30 (4), pp. 1-6.

Kintsch, E., Steinhart, D., Stahl, G. and the LSA Research Group. (2000). Developing Summarization Skills through the Use of LSA-Based Feedback. *Interactive Learning Environments,* Vol. 8, No.2. pp. 87-109

Klein, H., Myers, M. (1999). A set of principles for conducting and evaluating interpretive field studies. *MIS Quartely*, 23, p 67-93.

Komoski, P. K. (1987). Educational microcomputer software evaluation, in: J. Moonen, T. Plomp (Eds.), Eurit86: Developments in Educational Software and Courseware, Oxford: Pergamon Press, pp. 399-404.

Kvale,S. (1997). *Den kvalitativa forskningsintervjun*. Lund: Studentlitteratur. In Swedish

Landauer, T. K., Laham, D., Rehder, B., and Schreiner, M. E. (1997). How well can passage meaning be derived without using word order? A comparison of latent semantic analysis and humans. In Shafto, M. G. and Langley, P., editors, *Proceedings of the 19th annual meeting of the Cognitive Science Society*, pages 412–417. Mahwah, NJ. Erlbaum.

Laurillard, D. (1993). *Rethinking University Teaching. A framework for the effective use of educational technology*. London. Routledge

Löwgren, J. and Stolterman, E. (1998). *Design av informationsteknik*. Lund. Studentlitteratur. In Swedish.

Mayes, J.T. and Fowler, C.J. (1999). Learning technology and usability: a framework for understanding courseware. *Interacting with Computers, 11*, pp. 485-497.

Motschnig-Pitrik, R. and Holzinger, A. (2002). Student-centered teaching meets new media: Concept and case study. *Educational Technology & Society*, 5(4).

Nielsen, J. (1994). Usability methods, in: J. Nielsen, R.L. Mack (Eds.), *Usability Inspection Methods,* John Wiley, New York.

Norman, D.A. and Spohrer, J. C. (1996). Learner-centered education. *Communications of the ACM,* 39 (4), pp. 24-27.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York: Basic Books.

Patel, R., Tibelius, U. (1987). *Grundbok I forskningsmetodik.* Lund: Studentlitteratur. In Swedish

Person, N. K., Graesser, A. C., Kreuz, R. J., Pomeroy, V., and the Tutoring research group (2001). Simulating human tutor dialog moves in Autotutor. *International Journal of Artificial Intelligence in Education*, 12:23–39.

Resnick, M. (1997). *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. MIT Press.

Sklar, E. (2000). *CEL: A Framework for Enabling an Internet Learning Community*. Ph.D. thesis, Department of Computer Science, Brandeis University.

Sklar, E. and Pollack, J. (2000). An evolutionary approach to guiding students in an educational game. In *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior (SAB-2000)*.

Squires, D. and MacDougall, A. (1994). Choosing *and Using Educational Software: A teachers' Guide*. London: Falmer Press.

Squires, D. and Preece, J. (1999). Predicting quality in educational software: Evaluating for learning, usability and the synergy between them. *Interacting with computers, 11*, pp. 467-483.

Treiman, R., Introduction to special issue on spelling, *Reading and Writing: An Interdisciplinary Journal*, 9: 315–319, 1997.

Winship, J. (1988). Software review or evaluation: Are they both roses or is one a lemon?, In the *Proceedings of the Australian Computer Education Conference*, Perth.

Appendix I

Spell patterns used in all experiments.

	Pattern (regexp)	***Description***
1	.*[eyuioa]{2}.*	2v
2	.*[eo]{2}.*	eo2
3	.*ou.*	ou
4	.*iu.*	iu
5	.*oe.*	oe
6	.*oi.*	oi
7	.*io.*	io
8	.*ia.*	ia
9	.*ua.*	ua
10	.*ea.*	ea
11	.*au.*	au
12	.*eau.*	eau
13	.*[ei]{2}.*	ei2
14	.*ie.*	ie
15	.*ei.*	ei
16	.*[euyoai]?[st]e.{2,}"	
17	.*w[ndtl].*	w-
18	.*wn.*	wn
19	.*w[dt].*	wdt
20	.*wl.*	wl
21	.*rr.*|.*tt.*|.*pp.*|.*ss.*|.*dd.*|.*ff.*|.*gg.*|.*ll.*|.*bb.*|.*nn.*|.*mm.*|.*ck.*|.*cc.*	2-
22	.*rr.*	rr
23	.*tt.*	tt
24	.*pp.*	pp
25	.*ss.*	ss
26	.*dd.*	dd
27	.*ff.*	ff
28	.*gg.*	gg
29	.*ll.*	ll
30	.*bb.*	bb
31	.*nn.*	nn
32	.*mm.*	mm
33	.*cc.*	cc

	Pattern (regexp)	***Description***
34	.*ck.*	ck
35	.*[pbdw]t.*	-t
36	.*bt.*	bt
37	.*pt.*	pt
38	.*ct.*	ct
39	.*[wrtpgcs]h.*	-h
40	.*wh.*	wh
41	.*[st]?ch.*	ch
42	.*sh.*	sh
43	.*gh.*	gh
44	.+th.+	th
45	^th.*	|th
46	^kn.*	|kn
47	.*th$	th|
48	.*xc*.	xc
49	.*[sz].*|.*[aeiouyx]c[aeiouy].*	vow-c-vow
50	.*z.*	z
51	.*sc[aeiouy].*	sc
52	.*c[aoueiy].*	c
53	.*g[aoueiy].*	g
54	.*c[aou].*	hard-c
55	.*g[aou].*	hard-g
56	.*c[eiy].*	soft-c
57	.*g[eiy].*	soft-g
58	.*[iov]?e?s$	plur
59	.*[io]?e$	sing
60	.*[rtphgfdslkvbnm]y$	con-y|
61	.*[euoia]y$	vow-y|
62	.*[st]ion.*	-ion
63	.*[dbgkcp]le[sd]?$	-le
64	.*ing$	ing|
65	.*ng.*	ng
66	.*age$	age|
67	.*sed$	sed|
68	.*w$	w|

Appendix II

Questionnaire for interview with, schoolteacher in a grade 4 class.

Design: The spelling engine

1. How many years have you been teaching?
2. What levels have you been teaching at?
3. Have you got any teaching experience from other educational systems than the one in Queensland?
4. Comments on the design of The Spelling Engine?
5. What do you think of this kind of educational software?
6. Do you believe in constructivist method?
7. What statistics would be useful for you as a teacher to get from the system?
8. Would it be good to adjust the system so that a student almost every time will be successful with at least 5 out of 6 possible?
9. How can we encourage the students so that they will find spelling fun? (related to this program)

Appendix III

(This is the questions that were prepared for the more formal part of an interview. Informal discussions have been kept in a diary and then put together with the interview results.)

Questionnaire for interview of school teachers after use of the Magic Spell

Navigational fidelity: Is the user interface guiding or enticing or tricking the student to learn and hiding distracting complexity?

Do you believe the students learn how to spell using the Magic Spell?
Explain your belief.
Do you believe the program is appropriate for all grade three/four students?
If not, explain why.

Is there a match between designer and learner models? Does the user learn to spell as an effect of how the program teaches spelling? Is the feedback appropriate for all students irrespective of level? Does the design help learning?

Pedagogically, how do you think the students learn in the Magic Spell?
How frequently do you think you would have to use software like this to get the most out of it for learning?

Appropriate levels of learner control: Sense of ownership and control

Do you think there is sufficient feedback for students provided by the software?
Is the feedback appropriate for the age group?
Did you find the system picked an appropriate level of difficulty for each individual student?
Did you find that the students enjoyed working with the software? Provide arguments for your believe.

Prevention of peripheral cognitive errors

Was it good that students had to correct their own misspelling before they could move on?
Was there something in the design of the system that you found the students did not manage properly?
From the students' perspective, did you like the interface design of the system?
Was there anything you found hard to understand or easy to understand?

Personally significant approaches to learning: Does everyone irrespective of learning style learn how to spell efficiently?

From previous spelling exercises, have you identified different spelling learning styles in your class?
From using the Magic Spell, have you identified different spelling learning styles in your class?
Did you as a teacher find that the system supported students with different learning styles?

Was there a group of students you found the system unsuitable for?

Recognition and diagnosis, recovery from errors

Did you find that students understood what they had done when they did something wrong?

Did the system give students any indication to what was wrong when they made errors?

Match curriculum and teacher's customization

Do you find the Magic Spell useful as a complement to your teaching?

Do you find the Magic Spell useful for you students and you teaching goals?

Was it difficult to integrate the usage of the Magic Spell with other teaching activities?

Did you feel you had an influence on the design?

In relation to grade three/four curricula, is the content and process in the Magic Spell appropriate?

Do you have any suggestions of changes to improve the system?

Appendix IV

Questionnaire for interview of students after use of the Magic Spell

Was it fun to play the Magic Spell?
Why?

Do you think you became a better speller after using the Magic Spell?
Explain why?

Was it difficult or easy to spell?
Was it hard to win beans and wands?

Did you find it got harder to spell after working with the system for a while?
Did you like to trade wands with other students?

Was it easy or difficult to understand how to work with the Magic Spell?
Was there something that you found tricky when you started working with the Magic Spell?

Was it difficult to know where to click when you wanted to do something in the Magic Spell?
When you did something wrong in the Magic Spell, did you understand what you had done wrong?

Did the software indicate what you need to do to correct your mistake?