Design and Development of a Mathematical Teaching Aid for the Classroom

Mark Zachary
BSc in Computer Information Systems, 2004
Design and Development of a Mathematical Teaching Aid for the Classroom

Submitted by Mark Zachary

COPYRIGHT
Attention is drawn to the fact that copyright of this thesis rests with its author. The Intellectual Property Rights of the products produced as part of the project belong to the University of Bath (see http://www.bath.ac.uk/ordinances/#intelprop). This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the prior written consent of the author.

Declaration
This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Batchelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signature

........................................................................................................................................
Abstract

The aim of this project is to provide a set of requirements for an Information and Communications Technology (ICT) tutor to support the teaching of mathematics within primary school classrooms.

Research spanning a range of disciplines and empirical findings from the school environment are discussed. The requirements are then derived from this research, whereupon the design and development of a prototype to test these requirements is deliberated. A critique of this prototype is then completed in an attempt to extract additional usability findings from the domain. The implications of these findings upon future tutoring software are discussed within the conclusion.
Acknowledgements

I would like to thank my project tutor Hilary Johnson for her continual help and guidance throughout the duration of this project. My thanks also go out to Peter Wild for his help with various aspects of the evaluation.

This project was only possible thanks to the contributions of students and staff at Oldfield Park Junior School. Their accommodating attitude and keenness to offer support made the whole development process much easier.

This paper is dedicated to my family, and to all those across my childhood who have taught me the qualities of a good tutor.
# Contents

Section No. and Title:

**ABSTRACT** .......................................................................................................................... III

**ACKNOWLEDGEMENTS** ....................................................................................................... IV

1  **INTRODUCTION** .............................................................................................................. 1

2  **LITERATURE REVIEW** .................................................................................................... 2
  2.1 How can technology support learning? ................................................................. 3
  2.2 How do we learn? ........................................................................................................... 5
  2.3 An HCI Perspective ....................................................................................................... 6
  2.4 Summary ....................................................................................................................... 7

3  **REQUIREMENTS ANALYSIS & SPECIFICATION** ......................................................... 8
  3.1 Analysis ............................................................................................................................ 8
    Preliminary Classroom visits .......................................................................................... 8
    Examination of sample bookwork ................................................................................. 8
    Interviews with teachers, observations ........................................................................ 8
    Evaluation of existing ICT maths packages .................................................................. 9
    Findings from Participatory Design Approach ................................................................ 9
  3.2 Requirements Specification ........................................................................................... 9
    Functional Requirements ................................................................................................. 9
    User Requirements .......................................................................................................... 10
    Hardware Requirements ................................................................................................. 11
  3.3 Assumptions .................................................................................................................. 11
  3.4 Constraints .................................................................................................................... 12
  3.5 Conclusion ..................................................................................................................... 12

4  **DESIGN AND DEVELOPMENT** .................................................................................... 13
  4.1 Planning .......................................................................................................................... 13
  4.2 Evolution of the Prototype ............................................................................................ 13
    Initial (demonstration) Prototype .................................................................................... 13
    Secondary (pilot) Prototype ............................................................................................. 14
    Tertiary (and final) Prototype ........................................................................................ 14
  4.3 Software Design Discussion ........................................................................................ 14
    Choice of Implementation Language .............................................................................. 15
    Consideration of Theme and Characters ....................................................................... 15
    Storyboard ......................................................................................................................... 17
    Environment ..................................................................................................................... 17
    Navigation ........................................................................................................................ 18
    Reward System ................................................................................................................ 21
  4.4 Assessment of Prototype Against Requirements: ..................................................... 22
    User Requirements .......................................................................................................... 23
  4.5 Summary ......................................................................................................................... 24

5  **EVALUATION** ................................................................................................................ 25
  5.1 Planning .......................................................................................................................... 25
  5.2 Pilot Evaluation Session ................................................................................................. 25
  5.3 Tutorial Experiment ......................................................................................................... 26
  5.4 Results ................................................................................................................................ 26

6  **CONCLUSION** .............................................................................................................. 28
  6.1 Conclusion from the Results of the Experiments......................................................... 28
6.2 Overall Conclusion........................................................................ 28
6.3 Further Work.............................................................................. 29

7 REFERENCES..................................................................................31

8 APPENDICES..................................................................................32

A The Prototype Briefing Sheet ......................................................... 33
B The Final Prototype ........................................................................ 34
C Tutorial Transcripts ........................................................................ 40
D Assessment Paper .......................................................................... 45
E Assessment Results ........................................................................ 49
F Fun Toolkit Questionnaire .............................................................. 50
1 Introduction

Continual advances in technology have the potential to significantly expand the breadth and depth of the curriculum. Increases in the quality and availability of additional resources, such as online teaching material, provide a wealth of information far beyond the scope of traditional textbooks. Furthermore, the introduction of innovative support tools such as interactive whiteboards provide additional possibilities for the presentation of these resources. Despite a steady increase in the capability of technology however, many ICT implementations still fail to deliver. This paper sets out to develop and test a set of requirements to support future ICT developments.

The main objective of this paper is to develop a set of principles that allow for the design of robust and usable software-based tutors. When striving to engineer a comprehensive set of requirements, it’s necessary to adopt a user-centred approach (Zave 1997, Nusibeh & Eastbrook, 2000). This paper follows the approach embodied by Gould and Lewis (1985) – an early focus on users and tasks, empirical measurement throughout, and a cyclical development of a software prototype. The remaining sections of the document are presented in line with this approach. The second section of this report presents the theoretical research into the tutoring domain (section 2). Following on from this, a discussion of the empirical methods used to build an understanding of the user perspective (section 3). The results from both the theoretical and empirical research are analysed, leading to the generation of a set of requirements (section 4). The iterative development process adopted to shape the software is then considered (section 5), concluding with the contrasting of the prototype against the original requirements that seeded its content. Section 6 draws together the findings from all previous sections, and presents considerations for future work.
2 Literature Review

The application domain for the software discussed in this paper spans the fields of computer science, education and psychology, and initial consideration of the tutoring domain presents us with a series of fundamental questions tied to these disciplines. These fundamentals (presented in figure 2.1) are discussed below.

[Figure 2.1: Devised framework for literature review]

Support of teaching: A fundamental quality of the software defined by this paper will be its ability to support teaching. It is therefore important to study the teaching of mathematics and determine the methods by which an ICT tutor could best provide support. This project begins by examining both the structure and content of lessons in primary schools, as knowledge of these will allow for the easy integration of the software into existing environments. Whilst it is generally accepted that ICT tutors are a poor replacement for the real thing (Dwyer, 1974), a diverse range of opinions exists as to their effectiveness when implemented in support roles. For this reason, a digest of the arguments for and against the use of ICT within schools is presented for consideration. This research topic will be explored first, as it is felt that answers found in this area more than any other will narrow the scope of necessary research into the remaining fundamentals.

Learning Theory: Another issue that is fundamental to the success of any ICT implementation is the ease with which its lessons can be understood and stored as knowledge for future recall. With this in mind, examining current theory from the field of learning will build an understanding of the processes that the software
will need to support. This will be the second fundamental considered, as the consideration of a child’s capacity for cognition

The HCI Perspective: Through examining the psychology between the computer and the individual, the science of HCI has been able to accelerate the process of shaping technology to support human needs. Research within this field typically focuses on the need of adults, but some investigation should reveal those findings that can be applied to the domain of primary school education. Those HCI experts that specialise in catering for the needs of children will be of particular interest, and any existing heuristics from previous research will be examined and adopted if appropriate.

2.1 How can technology support learning?

The application of ICT to promote learning in schools is a topic that is greeted with very mixed feelings across the education community. Russel’s “No Significant Difference Phenomenon” (1999) evidenced that applying ICT to education rarely produced notable effects, and other sources confirm that technology implemented within systems often fails to meet expectation (Mann, Shakeshaft, Becker and Kottkamp, 1998). However, Christmann & Badgett (1999) present clear examples to the contrary, involving carefully considered implementations that have proven successful. Many of the problems with failed implementations seem to stem from the misconception that ICT is a solution in itself, when really it should be considered as part of a comprehensive, systemic plan to address specific challenges within education. (Van Melle et al., 2003).

Despite not being as versatile or comprehensive as a human teacher, the facility of an ICT tutor in a primary level classroom would be beneficial in a number of ways. Classroom teaching within schools typically features one teacher overseeing many pupils, and allocating time to a specific pupil for a length of time is often difficult when it is at the expense of managing the others. An ICT tutor would provide one-to-one guidance for pupils who needed help with concepts that they found particularly challenging. Conversely, the product could aid those students who were progressing faster than the rest. Either way, it could prove to be a useful additional resource if implemented correctly. In addition, students can repeat lessons that they find particularly challenging as many times as is necessary. This would be good for practise prior to assessments.

The implementation of ICT support aids has advantages and disadvantages over ordinary tuition methods. As mentioned previously, ICT tutors cannot rival a human tutor from a flexibility/resourcefulness perspective. A teacher’s lesson plans are much easier to amend than a software tutor’s build revision. Similarly, a human teacher will grow to appreciate the needs of any specific pupil in their class, but ICT tutors are far more restricted in their ability to “learn”, and existing examples of intelligent tutors have been known to make inappropriate judgements of the pupils’ skills and abilities, in the worst cases leaving them feeling either patronised or discouraged.
As set out in the national numeracy strategy, current resources in today’s primary schools do not allow for “a PC per student”, and so one probable eventuality will involve the teacher manipulating the completed software package in front of the class as a whole. It is possible however for individuals to work with it autonomously, and consideration is therefore needed towards the ease of navigation that children have around the product.

The National Numeracy Strategy (1999) provides a framework for the teaching of mathematics in primary schools that is adhered to across the country. This text is a manual for all numeracy lessons nationwide for children of all abilities up to Year 6 (typically the age of 11-12). There are structured syllabi for each of the year groups, and they are accompanied by exemplary lesson plans and suggested progression calendars for use across the academic year. These help to define the crucial principles of a particular syllabus that are fundamental to future progression. These syllabuses are based around Key Stages.

As discussed previously, the primary objective of this paper is to develop a set of requirements that can lead to the production of usable software tutors. In doing so, it will be necessary to produce a prototype. Resource limitations mean that the prototype can only test a limited number of objectives at most, and even then at only one particular key stage. This choice of key stage was at the discretion of a local primary school who suggested work aimed at the key stage 4 level (age 9-10).

Maths lessons in the primary classroom typically last between 45 and 60 minutes, and consist of:

- Oral work and mental calculation; the whole class work to rehearse, sharpen and develop their mental and oral skills
- The main teaching activity work as a class, individually or in pairs
- A plenary (to round off the lesson); the teacher irons out any misconceptions developed during the course of the lesson, identifies progress and reiterates the key facts, perhaps also setting homework.

From this framework, it is clear that tuition software supporting lessons would have to fit around this lesson plan. Content of the software tutorials could map directly to Key Stage learning objectives, and features that stretched beyond the length of the main teaching activity would be of little use in the classroom. Whilst this knowledge is helpful, it is important to remember that it is not necessarily set in stone. Software intended for use within education needs to be responsive to an environment that is often subjected to reform – changes in curriculum, time and space allocations are only to be expected (Honey et al., 1999).

Now that the level of material to be taught and the structure of the teaching has made clear, attention can turn to the methods in which people learn, and then

A specific subset of stakeholders has now been established, and the exploration of their needs will be considered in the next section.
2.2 How do we learn?

A broad range of theories exists on the subject of cognitive development. The behaviourist school believe that learning is a result of environmental stimulus – and that repeated exposure to specific stimuli results in the “learning” of behaviour (Pavlov’s dogs are a famous example supporting this theory). In opposition to this, constructivists - such as Jean Piaget - argue that knowledge is built from cognitive structures, or mental "maps" for understanding and responding to physical experiences within his or her environment. Piaget proposed a highly popular model for cognition that asserts that the structure of these models increase in sophistication as the child gets older, moving from simplistic reflex actions to complex cognitive abilities. His model categorises all children into four main stages of development based around their age:

- **The Sensorimotor stage (birth - 2 years old);** During this period, intelligence is demonstrated through motor activity without the use of symbols, although the foundation of symbolic manipulation is in place towards the end of this stage. The child, through physical interaction with his or her environment, builds a set of concepts about reality and how it works. This is the stage where a child does not know that physical objects remain in existence even when out of sight (object permanance).

- **Preoperational stage (ages 2-7)** Intelligence within this stage is demonstrated through the use of symbols. The child is not yet able to conceptualize abstractly and needs concrete physical situations.

- **Concrete operations (ages 7-11)** As physical experience accumulates, the child starts to conceptualize, creating logical structures that explain his or her physical experiences. Abstract problem solving is also possible at this stage. For example, arithmetic equations can be solved with numbers, not just with objects.

- **Formal operations (ages 11-15)** Nearing adolescence, intelligence is demonstrated through the logical use of symbols related to abstract concepts. Early in the period there is a return to egocentric thought. By this point, the child’s cognitive structures are like those of an adult and include conceptual reasoning.

Piaget’s work provided a framework for creating cognitive structures. During all development stages, the child experiences his or her environment using whatever mental maps he or she has constructed so far. If the experience is a repeated one, it fits easily - or is assimilated - into the child’s cognitive structure so that he or she maintains mental "equilibrium." If the experience is different or new, the child loses equilibrium, and alters his or her cognitive structure to accommodate the new conditions. This way, the child erects more and more adequate cognitive structures.

The age group of children defined in the previous section fall into the category of “concrete operations”. Consideration must be given therefore to the critical role
that experiences - or interactions with the surrounding environment - play in learning amongst this age group. For example, tutors have to take into account the role that fundamental concepts, such as the permanence of objects, play in establishing cognitive structures.

In conclusion, it now becomes clear that the cognitive development stage of the intended users will have implications on the requirements for the software tutor. In the next section, current theory from the field of HCI is explored in the hope that this will further clarify these fundamental considerations.

2.3 An HCI Perspective

As stated previously, user-centred design is fundamental to requirements engineering, but the involvement of children in software design requires careful contemplation. Whilst participatory design approaches can yield excellent results (Druin, 1999) excessive experimentation could prove counterproductive, as children could lose interest. Conversely, too little involvement may result in children feeling that their role is a superficial one (Preece et al., 2002).

Requirements elicitation can prove complicated when stakeholders themselves are uncertain as to their needs. Furthermore, it’s not unusual for the stakeholders of any project to present system requirements in a language quite unique to their field of expertise (Sommerville, 2001) and children are no exception to this rule. This might not be down to a lack of clarity on the issue, but could be because of their limited language and communication skills. Children, especially young ones, often have difficulty verbalising thoughts and ideas with clarity. As Janet Read (2003) stated, the phrase “I don’t know” from a child could mean anything from “I don’t want to tell you”, “I am too shy to answer”, “I perceive that you don’t want me to know the answer, so I’ll say I don’t know the answer”, “don’t know how to put it into words”, or plainly and simply that the child doesn’t know. Focusing on children aged 9-10 should mean that problems with articulation are few, but some shyness in certain individuals is to be expected.

As discussed previously, the software’s support of learning is one of the fundamentals of the domain. HCI methodology offers a range of possibilities for the assessment of software requirements. Druin’s recent participatory design workshops (2001) incorporate low-fidelity materials (such as paper, pens, crayons, furry felt, cloth, glue and scissors) in favour of complex technologies. Both adult and child assessors take notes on the interactions between the rest of the team as they work to create prototypes for software. These lab sessions see the emphasis shift from formality and towards fostering a spirit of creativity and fun. Although resource constraints on this project might make the recreation of such sessions unlikely, her findings can be brought onboard when attempting to elicit requirements. Druin also sets out a series of guidelines appropriate for working towards a participatory design with children, from simple tips (Dress informally, sit where possible in favour of standing, ask their opinions and allow plenty of time for the formulation of responses) to more regimented guidelines for
gathering empirical information. These guidelines will be examined in more detail in the following section.

A better understanding of the user perspective could come from interviewing teachers and pupils. This might help foster an understanding of their opinions and any problems that they have with working with existing ICT tools.

Principles of good design prescribe placing the user at the forefront of the design process. With this in mind, the first question to be addressed will be “Who are the stakeholders for this product?”

**2.4 Summary**

Fundamental questions relating to existing teaching methods have given us an insight into the structure and content of mathematics lessons in schools. It emerged that software that is both flexible and robust will be essential in an environment that is prone to rapid change. Some investigation into how these attributes can be engineered into software is needed.

Popular theory from the field of developing cognition was explored, and the characteristics typical of those children in the concrete operations phase of cognitive development were considered. The remainder of this literature survey explored the HCI perspective, and possible barriers to feedback presented by shyness, uncertainty and people-pleasing behaviour in children. Appropriate heuristics that can provide guidance when attempting to design and develop software for children were touched on briefly, and they will be discussed further in the following chapter. It is hoped that their findings will help to develop a richer set of initial requirements from which the prototype will be built. Exemplary feedback media from such assessments will be attached within the appendices.
3 Requirements Analysis & Specification

Studying the intended application domain for the software will allow us to identify more clearly the needs of the stakeholders. Consideration of the stakeholder perspective was touched upon in the literature survey, as the intended age range of children that the software will be targeted at was established. It is important to remember that the pupils are not the only stakeholders to consider. The teachers, learning assistants and even school governors are also stakeholders. The findings from empirical research – exploring the needs of stakeholders - are presented here.

3.1 Analysis

Analysis of empirical evidence was completed in line with the framework produced by Allison Druin (1999). Various research activities were completed that contributed towards the empirical work. These were:

- Preliminary visits to classrooms – general teaching observation
- Interviews with teachers and pupils
- Examination of sample bookwork
- Evaluation of existing ICT maths packages
- Findings from participatory design approach

Each of these is discussed in turn.

Preliminary Classroom visits

Prior to conducting any interviews with teachers or pupils, volunteer work was conducted in a local primary school. This allowed for research into the interests of children between the ages of 9-10, an insight into their reading age (the next J.K Rowling novel was eagerly awaited – a healthy reading age) and also typical maths problems as they were presented in class.

Examination of sample bookwork

A textbook containing problems typically solved by children approaching Key Stage 4 competency was procured, and the sample questions were noted alongside those supplied in the National Numeracy Strategy.

Interviews with teachers, observations

The textbook work mentioned in the previous section was discussed with a panel of teachers, which resulted in deliberation over a suitable level of activities for
inclusion within the software. The eventual outcome of these discussions fed into the requirements as well as the design of the prototype (see section 4).

**Evaluation of existing ICT maths packages**

Informal evaluations were conducted of some standalone packages intended for use outside the classroom, or as fun activities for users that were ahead of the rest of the class. Some were produced in accordance with Key Stage Learning objectives, whilst others were far less structured.

**Findings from Participatory Design Approach**

These findings are presented in the next chapter of the paper.

### 3.2 Requirements Specification

The section details the requirements produced through the methods outlined previously. The requirements have been categorised by whether they relate to functional aspects of the interface, hardware requisites or user-guided principles. Within these categories, they are ordered by their perceived importance (most important first). Each requirement is coupled with the rationale that led to its generation.

**Functional Requirements**

Functional requirements are those that relate to requisites from the software itself.

**Requirement:** The software should support the teaching of mathematics – with particular reference to the structure and content dictated by the national numeracy strategy

**Source:** Following on from the fundamental questions explored in the literature review, it is important that this structure is followed as closely as possible.

**Requirement:** The software should allow for flexibility and modification for the specific school or class’s need.

**Source:** Derived from the previous requirement, that fully working software should map perfectly onto the national numeracy strategy content. The comprehensive software should support configuration (e.g. Welsh language).

**Requirement:** The interface should be straightforward and intuitive for use by both children and adults alike.

**Source:** From the notion of affordance discussed within the HCI perspective section of the literature review.
**Requirement:** The software should feature virtual personalities that children can interact with.

**Source:** This relates to research into the study of Piaget’s work. Piaget (1970) asserted that some children in the concrete operations phase of development had trouble understanding number problems without a context to apply them to. The example given was that conservation of number proved problematic until a character - “naughty teddy” – was introduced to simplify the problem.

**Requirement:** The software should allow for the endless repetition of exercises that children want to practise, and furthermore, there should be no negative feedback to this desire for repetition

**Source:** From the behaviourist school of thought towards learning that suggests “drill and practice” leads to the absorption of new knowledge.

**Requirement:** The software should include a help feature to support users who encounter problems

**Source:** Research into existing ICT tutors noted that all packages examined contained a help feature of some descript.

**Requirement:** The software should provide some facility for customisation of fonts and sounds to allow for the needs of pupils with learning difficulties.

**Source:** Research into existing ICT tutors noted that all packages featured some method of modifying the difficulty and length of exercises.

**Requirement:** The software should feature a progression and reward system

**Source:** The ICT packages examined all featured reward systems.

**Requirement:** The program should allow the user to add, view, edit, delete and store user data.

**Source:** Several of the ICT packages observed featured save and load facilities.

**User Requirements**

These are non-functional requirements relating to essential development considerations to provide support for the end user.

**Requirement:** The generation of software prototypes should be an iterative process, with feedback from users inputted into subsequent versions.

**Source:** This requirement stems from the decision to adopt Gould and Lewis’ (1985) model of a user centred approach as outlined in the introduction. It is
further supported by research into HCI theory which highlights participatory design as a popular method of design with children (Druin’s participatory design workshops)

**Requirement:** *The software should reflect the interests of its target audience*

**Source:** This requirement stems from the previous one. Participatory design is a method of improving the chance that the software will meet user expectation.

**Requirement:** *The software should strive to maintain an informal level, in the same way as that shown in games.*

**Source:** Taken from consideration of the HCI perspective from within the literature review. Not all of the design principles that apply to adult software apply to software designed for children.

**Hardware Requirements**

These are requirements which are tied directly to hardware capabilities of the system.

**Requirement:** *Suitable means of storage should be provided*

**Source:** There was no research source for this requirement.

**Requirement:** *The software should run quickly and smoothly*

**Source:** There was no research source for this requirement.

### 3.3 Assumptions

- It should be assumed that the completed software product is intended for use as a teaching aid in class – that is - to assist the existing teaching methods currently in practise across schools nationwide. It is by no means intended as a replacement for a teacher, or to be used as an independent resource outside lessons.
- The target technology is the present day minimum specification upwards, such that most people would be able to run the program on their machines.
- It will be assumed that the target audience is a single language audience, typically English.
3.4 Constraints

- Whilst the requirements outlined in this section relate to children of all ages, exposure to children is reasonably restricted...

- Due to time constraints on the project the number of tutorials and analysis methods that can be run in order to best understand the problem will be limited. This also poses a constraint on the range of participants that can be tested.

3.5 Conclusion

The empirical and theoretical knowledge gathered in the previous two chapters has been examined, and findings have been turned into requirements listed in this section. It remains to be seen whether software built in line with these requirements really does support learning, so the next section details the planning and execution of various evaluations in order to assess the usability and efficacy of the prototype tutor as well as presenting the development and design decisions that were made when creating it.
4 Design and Development

This section discusses the decisions to be made during the creation of the incremental software prototypes. Focus centres on designing functionality to support the requirements drawn from the literature.

4.1 Planning

It should by now be clear to see that the design that is created is intended to map directly from the requirements in the previous chapter. The content of the prototype was devised with the help of teachers from a local school to be appropriate to material currently being taught within maths lessons.

The national numeracy strategy features exemplary questions, and a few of these were selected and a mock-up of possible interactive games to feature in the prototype. As discussed previously, time and resource constraints do not permit for an entire suite of tutoring functions to be developed, and discussion of the precise content of the prototype will be considered toward the end of this section.

4.2 Evolution of the Prototype

In accordance with the participatory design technique proposed in the requirements, the prototype was evolved through cyclic reviews with a panel of teachers and children. At each stage the minimum amount necessary was implemented and then the prototype was evaluated and any new considerations incorporated into the refined design. In total there were three main software versions, the major features of which are discussed below.

Initial (demonstration) Prototype

The first build of the software provided the opportunity to build up a knowledge of the implementation language and environment.

The first prototype was simplistic, with little functionality. It was simply a glorified storyboard with simple animations, and very little action script featured within the demonstration prototype. As specified in the requirements section, the completed software would provide coverage for the major aspects of the five key areas in Key Stage 4, but resource restrictions meant that focus was limited to a single feature. Work began on creating puzzles to test mathematical ability. Several basic demonstrative puzzles were created, and one was selected based on feedback from the teachers of a local primary school. A panel of teachers, consisting of the school’s ICT coordinator, head of mathematics and the deputy headmistress, screened the initial prototype. More informally, several fellow
undergraduates examined the software and offered their hints and suggestions for improvements.

Following on from the feedback received from schools, significant portions of this initial prototype had to be scrapped. However, the intended focus of the game was decided as a result of this feedback, and work began on the second prototype.

**Secondary (pilot) Prototype**

Having decided upon a particular activity to focus upon as the main feature of the prototype, work began on completing the second prototype. This version was created almost from scratch. The same graphics were used in the second prototype, but they were cleaned up using graphical smoothing techniques to improve their appearance and give a more professional feeling.

Originally, an attempt was made to seed a random number generator from the system clock, but that proved too cumbersome. The “Math.random” function provided the necessary random numbers to seed the randomly generated road signs.

In order to ensure that the software struck a balance between being too easy and too challenging, an incremental difficulty level was implemented. This feature allowed different difficulty levels to be selected depending on the pupils’ ability, including pupils within the same age range.

This second prototype was used during the software pilot. The level of questions were broader than those in the final submission, with hundreds, tens and units, as well as different measurement units (centimetres, metres and kilometres), being taken into account. Following on from the results of the pilot (discussed fully in chapter 5), a third prototype was engineered.

**Tertiary (and final) Prototype**

The third prototype was a smoother running and more polished version of the secondary build. Comments and opinions from the children who took part in the pilot test were taken on board, and the kilometres conversion questions were removed on the advice of teachers who thought it too challenging for children at a key stage 4 level.

Following a brief review of the software from the same panel of teachers, the software was evaluated using the method described in chapter 5.

### 4.3 Software Design Discussion

Reasoning behind the design decisions is presented below.
Choice of Implementation Language

The constraints upon this project that have been previously outlined coupled with the decision to adopt a participatory design stance considerably narrow the possibilities for a chosen implementation language. The language needs to be well suited to the task of generating high-quality prototypes with animations and sound. Macromedia Flash - typically implemented as a high-level web design tool - allows such prototypes to be generated very quickly, and has a scripting language integrated within itself known as action script. This scripting language is very similar to JavaScript. Consideration was briefly given to both Java and C++, as the author had reasonable experience coding in both of these, but their use would have necessitated large libraries of functions that would have taken too long to prepare, and effort in their preparation would be wasted if the resultant prototypes were then discarded. Comparatively fast prototyping and impressive graphical results made Macromedia Flash the most appropriate tool to use.

Consideration of Theme and Characters

The elected theme for the software prototype was a result of the preliminary visits to schools, interviews with pupils and a little wider reading. Recent cinematic interpretations of J.R.R. Tolkien’s books coupled with the enduring appeal of the works of J.K. Rowling have meant that the fantasy theme is a prominent and popular one amongst children today. Additional confirmation of this came from the web site of the children’s retailer’s Hamley’s, who compile an annual list of their best-selling books & toys. Consultation of this list revealed several distinct fantasy-based entrants.

It was clear from the requirements that a virtual character was needed as a protagonist. Preece (2002) discusses the conflict between realism and abstraction in virtual characters. Abstraction was the obvious choice, as the interface was designed with children in mind. In addition to an identity, the protagonist obviously needed a cause – a reason for setting out on an adventure filled with mathematical puzzles. Knights seem to go hand in hand with dragons, and so Sir Eric and Johina the dragon were created.
Thick black outlines around the major objects in the foreground gave things definition and a “cartoon” feeling. This was to highlight major characters or objects that would become interactive at some stage. (In addition, the knight’s helmet neatly sidestepped the matter of complicated mouth animations that would have been fairly time consuming on a project of limited resource.)

More angular shapes, and the use of dark backgrounds on the arrival of Johina on screen were used to give the dragon graphic the aura of villainy. (More example screen shots can be seen in Appendix B).

Two of the main objects featured were the signpost and the catapult. (The catapult was removed from the later prototypes, as it was decided that the signpost exercise should be the sole activity evaluated)
Storyboard

Presented here is one of the low-fidelity prototypes constructed prior to the second prototype build. This model was presented to teachers within the school, and enabled after some discussion, it was decided that the storyboard be amended to loop around and pose the same question again in the event of an incorrect answer.

From here, consideration was given to the setting of the navigation within the software - the methods by which the users would explore the prototype.

Environment

Use of colour, use of sound: This was important

Preece et al. (1994) stress the importance of colour as a tool for highlighting or dividing up a display. And this was adhered to when creating a “main stage”, notably the clear distinction between the foreground, hills and sky. – typical usability guidelines relating to the “overloading” with too many colours obviously apply – but then again, this is not a professional company web site or piece of software that is being developed, it is a fun learning tool for children.

Consideration was given to striking a balance with the audio instructions. The instructions needed to be slow and clearly spoken so that the children could easily understand how to use the software and follow the voice prompts. It was very
important that they were not deterred by complex sets of instructions, or flummoxed by quick speech. However, speech needed to be sufficiently fast so as to not patronise them.

Simple animations were made with ease using Flash’s motion tweening feature. This allowed a series of similar still graphics to be cycled in order to create the illusion of motion much like in a flick book. Three frames were used that made protagonist “Sir Eric” move (see figure 5.3.1.1)

![Figure 4.3.1: Sir Eric stationary and in motion]

**Navigation**

As can be seen from the storyboard presented earlier, navigation through the software is fairly linear, with only one scene offering the user a choice of options for progression (the puzzle scene - see frame 3 of fig 4.3.2.1)

With the exception of the puzzle frame (frame 3), chevron buttons were used to progress to the next screen (see fig. 4.3.4.1). The button’s purpose should be intuitive, and it was hoped that the sharp contrast between the bright red of the button and the green of the grass would draw it to the user’s attention. The dual arrows are commonly used to indicate “forward” in everything from VCRs to stereo systems to indicate forward motion, so this shape was adopted so that it might afford pushing.

![Figure 4.3.4.1: (l-r) “up” and “over” button states]

The puzzle scene (frame 3 in the storyboard) was unique, as it was the only scene that allowed the user a choice of inputs, and navigation was through clicking upon the road signs themselves (fig. 4.3.4.2). The entirety of each road
Design and Development of a Mathematical Teaching Aid for the Classroom

sign was a hot spot, and the signs were designed with tapered ends to suggest progression to the user.

![Figure 4.3.4.2: Typical roadsign hot spot]

The numbers on the signs were randomly generated between appropriate bounds using a short script that took a random number and processed it according to the selected difficulty setting (and, in the case of difficulty setting 3, presented the user with a mixed-units question 3 times out of 5).

```java
if (lastAnswerCorrect == 1){
    currentQuestion = currentQuestion + 1;
    option1 = 0;

    while ((option1 < 0.1501) || (option1 > 0.9498)) {
        option1 = /*Math.floor*/Math.random();
    }

    if (difficultySetting == 1){
        option1 = Math.floor(option1 * 10000);
    }
    if (difficultySetting == 2){
        option1 = (Math.floor(option1 * 10000)/10);
    }
    if (difficultySetting == 3){
        option1 = (Math.floor(option1 * 10000)/10);
    }

    shortestDistance = Math.floor((Math.random())*3)+1;
    if (shortestDistance == 1) {
        option3 = option1;
        option1 = option3-(Math.floor(option1/10)-1);
    }
```
option2 = option3+(Math.floor(option1/10)+1);
}
else if (shortestDistance == 2) {
    option2 = option1-(Math.floor(option1/10)-1);
    option3 = option1+(Math.floor(option1/10)+1);
}
else if (shortestDistance == 3) {
    option2 = option1+(Math.floor(option1/10)+1);
    option3 = option1-(Math.floor(option1/10)-1);
}

units1 = "m";
units2 = "m";
units3 = "m";

if(difficultySetting == 3){
    unitChange = Math.floor((Math.random())*5)+1;
    switch(unitChange){
        case 1:
            option1 = Math.floor(option1);
            option2 = Math.floor(option2/10)/10;
            option3 = Math.floor(option3/10)/10;
            units1 = "cm";
            break;

        case 2:
            option2 = Math.floor(option2);
            option1 = Math.floor(option1/10)/10;
            option3 = Math.floor(option3/10)/10;
            units2 = "cm";
            break;

        case 3:
            option3 = Math.floor(option3);
            break;
    }
}
Design and Development of a Mathematical Teaching Aid for the Classroom

```javascript
option1 = Math.floor(option1/10)/10;
option2 = Math.floor(option2/10)/10;
units3 = "cm";
break;

case 4:
    option1 = option1 * 10;
    option2 = option2 * 10;
    option3 = option3 * 10;
    break;

    case 5:
        break;
}
}
}
}
stop();
```

[Figure 4.3.4.3: Sample action script to generate random distances]

**Reward System**

A series of icons were used to signify correct and incorrect answers to questions. The system only recorded the first answer given for each question, so getting it right the second time around wouldn't credit the user’s score. It was considered that the score screen (frame 6) should feature different animations dependant on the number of questions answered, but this proved too much to implement within the time constraints of the project.

![Reward System Icons]

[Figure 5.3.1.1: Sir Eric (signifying a correct answer and Johina embodying an incorrect answer)]
4.4 Assessment of Prototype Against Requirements:

In an attempt to gauge the extent to which the software meets its intended targets, the prototype will be set against the requirements listed in the previous section of the paper. This will be the final assessment prior to the execution of the experiments in the following chapter. (NB: the hardware requirements aren’t included here, as they aren’t in anyway related to the features of the software.)

**Requirement:** The software should support the teaching of mathematics – with particular reference to the structure and content dictated by the national numeracy strategy

**How the software works:** The software prototype’s one puzzle was designed based on exemplary questions from textbooks and the feedback from a panel of teachers.

**How this action meets this requirement:** It was felt that the prototype met this requirement adequately.

**Requirement:** The software should allow for flexibility and modification for the specific school or class’s need.

**How the software works:** There is not currently any support for modification to the prototype bar the ability to increase/decrease the difficulty of the exercise.

**How this action meets this requirement:** This requirement was only partially met.

**Requirement:** The interface should be straightforward and intuitive for use by both children and adults alike.

**How the software works:** The interface adhered to guidelines on colour, opted for bold graphics and a navigation system that adhered to the principle of affordance.

**How this action meets this requirement:** This requirement was met adequately.

**Requirement:** The software should feature virtual personalities that children can interact with.

**How the software works:** The protagonist character within the game is an abstract virtual personality, and basic animations (“tweens”) were used to make this personality more lively.

**How this action meets this requirement:** This requirement was met adequately.
Requirement: The software should allow for the endless repetition of exercises that children want to practise, and furthermore, there should be no negative feedback to this desire for repetition

How the software works: The software supports endless repetition of the main exercise.

How this action meets this requirement: This requirement is met completely.

Requirement: The software should include a help feature to support users who encounter problems

How the software works: Currently no help features were implemented within the software itself. A briefing sheet has been devised for use with the prototype (see Appendix A).

How this action meets this requirement: This requirement has not been met.

Requirement: The software should provide some facility for customisation of fonts and sounds to allow for the needs of pupils with learning difficulties.

How the software works: No support for children with learning difficulties is currently provided.

How this action meets this requirement: This requirement has not been met, but this is something difficult to incorporate into software with such restricted resources.

Requirement: The software should feature a progression and reward system

How the software works: The prototype software features icons that display ticks or crosses when the user answers questions correctly or incorrectly

How this action meets this requirement: This requirement has been met completely.

Requirement: The program should allow the user to add, view, edit, delete and store data.

How the software works: No functionality is currently incorporated to support storage of results

How this action meets this requirement: This requirement hasn’t been supported.

User Requirements

Requirement: The generation of software prototypes should be an iterative process, with feedback from users inputted into subsequent versions.
**How the software works:** An iterative process was adopted, and the three prototypes used have been supplied upon the CD enclosed with this paper.

**How this action meets this requirement:** This requirement has been met completely.

**Requirement:** The software should reflect the interests of its target audience

**How the software works:** Particular effort was put into researching an appropriate theme for the software. The initial feedback to the design was encouraging indeed.

**How this action meets this requirement:** This is a hard requirement to quantify, but it was felt that it had been met adequately.

**Requirement:** The software should strive to maintain an informal level, in the same way as that shown in games.

**How the software works:** The jocular cartoons and friendly interface were created with informality in mind.

**How this action meets this requirement:** Initial user feedback was encouraging, one user even gave the feedback “much more fun than working in books!” The author feels that this certifies

4.5 **Summary**

The design and development of the software prototype has been presented in this section. Consideration was given to the underlying theme, the characters, the progression of the game, In-game navigation and the implementation of a reward scheme. The final implementation has been assessed against the requirements that it set out to adhere to. Building on these findings, it is now necessary to find some method of assessing the software’s ability to teach.
5 Evaluation

Following on from the development of the prototype, it was necessary to evaluate its effectiveness at tutoring. The following sections present the planning, execution and results of the evaluation sessions. Transcripts of tutorials and a copy of the briefing sheet and test can be found in the appendices.

5.1 Planning

The software prototype was initially presented to a small group of students and used in tandem with a paper-based mock up of the software as part of a pilot experiment. The findings from the pilot coupled with feedback from experts led to the development of a final prototype (discussed in the previous section), which was used to tutor two groups of six students on the topic of place value incorporating measures.

Prior to the full evaluation, a brief pilot test was carried out on a group of four students of mixed ability. This test was carried out with participants from a local school, this involved n kids experimenting with the software and giving feedback on the form and The assessment form was designed based around Read’s Fun toolkit (see Appendix F) form for gauging assessment. This was done in the hope that some feedback on the usability of the product could be gauged.

This pilot was designed in order to establish precisely which difficulty level would be most appropriate for use within the completed prototype.

Following on from the pilot test, a series of tuition sessions were completed on a set of 14 pupils. Participants were tutored on a one-to-one basis, and the sessions had a cap of 20 minutes, as it was felt that taking them out of class for longer than this would be disruptive to their schoolwork.

Exemplary transcripts of tuition sessions can be found within the Appendix.

5.2 Pilot Evaluation Session

A pilot evaluation session was completed prior to the full evaluation of the software. The second version of the software prototype was used in tandem with a paper-based version. This paper-based version (Appendix B) was to act as a control measure to allow for comparison between standard tuition and tuition assisted by the ICT software.

As discussed in the previous chapter, the pilot software featured a very different set of problems to those in the final prototype (more complex sums involving cm and km), and this wider range of difficulties was intentional. Questions were included that fell below and rose considerably above the minimum key objectives for key stage 4. This was done in order to gauge the upper and lower echelons of ability within the pilot group.
The results from the pilot evaluation were varied indeed; some of the children seemed happy with working with all three sets of units, whilst other did not. The huge range in ability, particularly mathematical ability, at this age range was highlighted through these experiments, but this was not altogether unexpected.

In light of the findings from the pilot, it was decided that questions involving kilometre-conversion problems would be removed from the prototype, and those involving centimetres would have to be significantly simplified, as they proved too taxing for the majority of pupils that took part in the trial.

5.3 Tutorial Experiment

In preparation for the evaluation, a group of six students were led through a twenty-minute introduction (one to one tuition). Transcripts from these tutorial experiments can be found within the appendix (section C).

As a control step, a second group was given the same tuition with the same briefing before each tuition session (see Appendix A). This tuition was carried out using a paper-based version of the software. The paper-based version of the software consisted of screenshots from various puzzle questions in the game.

The test content was modelled upon the style of question presented within both tuition sessions, and an extra few questions were asked in a similar style to those encountered in the Collins textbooks that were used within the school.

5.4 Results

A copy of the test completed by both groups of participants, as well as a complete breakdown of the results collated, can be found in appendix E. For convenience, the overall percentages of the twelve complete entries are presented here. Of the fourteen students that originally took part in the survey, one was unable to take part the due to absence, whilst another failed to complete any of the questions within the allocated time period.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Gender</th>
<th>Percentage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Female</td>
<td>88</td>
</tr>
<tr>
<td>A2</td>
<td>Male</td>
<td>81</td>
</tr>
<tr>
<td>A3</td>
<td>Male</td>
<td>69</td>
</tr>
<tr>
<td>A4</td>
<td>Female</td>
<td>13</td>
</tr>
<tr>
<td>A5</td>
<td>Male</td>
<td>44</td>
</tr>
<tr>
<td>A6</td>
<td>Male</td>
<td>44</td>
</tr>
</tbody>
</table>
Table B: Percentage Scores of Pupils who used the Paper Based Prototype

<table>
<thead>
<tr>
<th>Participant</th>
<th>Gender</th>
<th>Percentage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>Male</td>
<td>100</td>
</tr>
<tr>
<td>B2</td>
<td>Female</td>
<td>63</td>
</tr>
<tr>
<td>B3</td>
<td>Male</td>
<td>50</td>
</tr>
<tr>
<td>B4</td>
<td>Female</td>
<td>31</td>
</tr>
<tr>
<td>B5</td>
<td>Male</td>
<td>69</td>
</tr>
<tr>
<td>B6</td>
<td>Male</td>
<td>25</td>
</tr>
</tbody>
</table>
6 Conclusion

This paper set out to explore the factors affecting the success and failure of ICT implementations within schools. After conducting a literature survey and gathering a broad range of empirical data, the requirements were drawn up, and a prototype was developed. Experiments were then carried out using this prototype in a primary school maths class in order to determine how it performed, and what improvements could be made with further development and design changes.

6.1 Conclusion from the Results of the Experiments

It was difficult to draw definitive conclusions from the results of the experiments. Overall, those pupils who carried out the experiment using the software obtained slightly better results than those who used the paper based mock-ups. However, resource and time restrictions meant that the experiments could not be carried out with a larger number of pupils, which would show more conclusive evidence as to whether this was continually the case.

One possible reason for such results could be that pupils are more familiar with looking at mathematical problems on paper, and so are not as stimulated by them as seeing an animated, colourful piece of software on a screen in front of them that captures their attention and requires their input. However, all of those pupils involved in the evaluation had substantial experience with using computers, and they were unified in their enthusiasm for taking part in the evaluation sessions (“it’s much more fun than working in books!” was one comment) so this theory is unlikely. Whilst the results of the evaluation are themselves inconclusive, the students seemed to enjoy taking an active involvement in the process of designing and assessing the software prototype.

The decision made earlier in this chapter to evaluate using students with a wide range of abilities makes it somewhat difficult to establish the cause of the weaker test results. It’s possible that the children with lower marks had less developed cognitive abilities than those of their peers but without the resource to investigate these suggestions further this is purely speculative. A much large number of participants and results would be needed to drawn a more definitive conclusion.

6.2 Overall Conclusion

This project has sought to answer a series of questions about how children learn and how ICT can best cater for their needs within the field of maths.

The prototype developed was certainly embraced enthusiastically by the participants, and it would be fair to say that the software successfully engineered “fun” where it may have failed to engineer usability. This in itself is encouraging,
as if children perceive an activity to be fun, they are more likely to become more involved in the lesson, and will want to return to the software or similar systems in the future. The requirements established in section 4 were a useful guide to prototype creation, but the inconclusive results from evaluation suggest that more testing is needed. Greater resources would allow for the widening the number of learning tools and methods is also very effective, as pupils have a greater variety of resources and would therefore not become bored as easily as they may do otherwise.

Within the field of maths, there is a wide scope for many different types of tools that could be developed and used to aid teaching and increase the learning experience for pupils. From the development of the software in this project in various different stages, the key functionality that was required an example of such a tool was determined further with each prototype, leading to a better product overall that had addressed the issues raised during its design and development.

6.3 Further Work

Allowing for additional time and resources, further development work that this project could take would include some of the ideas of extensions that are detailed below.

A larger number of experiments could have been carried out with pupils at different age ranges, had further time and the availability of more pupils been permitted. These experiments could be used to determine more accurate difficulty levels that would be appropriate for different age ranges, or within a particular age range.

However, the experiments that were carried out indicated such a broad range of abilities of pupils who were the same age that it would only be possible to determine a recommended difficulty level for a particular age group, with the option of selecting a different ability level incorporated into the software, as it was done.

Use of feedback from both teachers and pupils was done when developing the prototype further, with particular issues relating to different aspects of the software addressed. This could be carried out in more depth with further feedback from those who participated in the experiments and from additional feedback from a larger number of users.

These issues could include the design of the interface, colour, usability from the pupils’ point of view and from the teachers’ point of view in terms of incorporating the product into lessons, and whether both groups felt it was a useful in terms of actual learning.

This information could be collected from the teachers and pupils via further questionnaires, but this again raises issues of time and resources, and the pupils may feel it takes some of the fun out of using the product if they are asked a large number of questions relating to it – asking them whether they enjoyed using the software and what they particularly liked or didn’t like about it may be as far as it is practical to ask.
Further development work and re-design of the prototype could then be carried out to make the software address as many of the issues raised as possible, thereby increasing its appeal to be used in conjunction with more traditional learning and teaching methods.

Research and use of statistical methods could be carried out, which could then be used to analyse the results of the experiments further. Comparisons could then be made to include testing hypotheses to determine whether the software prototype or the paper-based experiments produced higher results from pupils. However, to gain sufficient proof that one method did continually produce higher results from pupils, a far larger number of participants would have to take part in the experiments.
7 References


Russell, T. R. (1999) "The no significant difference phenomenon"


8 Appendices

A THE PROTOTYPE BRIEFING SHEET.........................................................33
B THE FINAL PROTOTYPE........................................................................34
C TUTORIAL TRANSCRIPTS....................................................................40
D ASSESSMENT PAPER...........................................................................45
E ASSESSMENT RESULTS ......................................................................49
F FUN TOOLKIT QUESTIONNAIRE ..........................................................50
THE STORY
Sir Eric the Unready set out on a quest this morning, but on his way home, he was spotted by Johina the dragon! Within this game, you must guide Sir Eric home to the safety of his castle before Johina can catch him.

PLAYING THE GAME
To reach the castle as quickly as possible, you must choose the shortest path from each of the road signs along the way. Most of the road signs across the kingdom present distances in metres, but some give distances in centimetres. This conversion table will help you translate these figures into metre units.

| 100cm = 1 metre | 600cm = 6 metres |
| 200cm = 2 metres | 700cm = 7 metres |
| 300cm = 3 metres | 800cm = 8 metres |
| 400cm = 4 metres | 900cm = 9 metres |
| 500cm = 5 metres | 1000cm = 10 metres |

Sometimes the distances displayed on the signs won’t translate perfectly into whole metres, so you must be careful! Use these examples to help you:

| 50cm = 0.5 metres |
| 120cm = 1.2 metres |
| 330cm = 3.3 metres |
The Final Prototype

The software CD attached at the rear of this document features the source files in .fla format as well as in several executable file formats.

The following set of images was used as a paper-based prototype for the tuition of place value and measures to the control participants. Each image is an edited screenshot taken from the software prototype set at difficulty level 3.

Problem 1:

Problem 2:
Problem 3:

Problem 4:
Problem 5:
Problem 6:

Problem 7:
Problem 8:
Appendices

Problem 9:

Problem 10:
C Tutorial Transcripts

As described in the main body of the paper, the proceedings of the tutorials were recorded for careful study. The experiments yielded two particularly responsive participants, whose high levels of extroversion meant that they were keen to give feedback on most aspects of the activity and the prototype itself.

Transcripts of the tutorials with these two more extroverted participants are presented in this section. Important gestures and actions carried out are noted in square brackets. The question that the transcript relates on the screen at each stage is denoted in curved brackets.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Role</th>
<th>Prototype</th>
<th>Age</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Tutor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>Participant</td>
<td>Software</td>
<td>9</td>
<td>Male</td>
</tr>
<tr>
<td>B1</td>
<td>Participant</td>
<td>Paper</td>
<td>9</td>
<td>Female</td>
</tr>
</tbody>
</table>

**Transcript of Participant A1**

[The plot of the game is explained using the briefing sheet (Appendix A). The software is started up, and the participant assumes control of the interface.]

1> **A1**: [clicks on the “continue” button in bottom right corner of first screen. Listens to audio instructions and then clicks continue to begin game. The first set of questions appears on the screen.]

2> **T**: Right! So here are the first few questions... Which of these distances do you think is the smallest?

3> **A1**: That one? [Points and clicks on the correct answer, an animation plays confirming the right choice has been made. Presses “continue”.]

4> **T**: That’s right! Well done! Now keep going, and talk me through each problem...

5> **A1**: [The second question appears. The Participant considers the question, hovers the cursor over 182.8, once again identifying the correct answer.] hmmm... That one?

6> **T**: You think that that’s the smallest of the three, yeah?

7> **A1**: Yeah. [Clicks the answer onscreen]

8> **T**: That’s right! Well done!
A1: [Participant considers the third problem. The screen displays two distances in metres and a third in centimetres] We have a distance in centimetres, now do you know the difference between metres and centimetres?

T: Well, let’s look back to this sheet... How many centimetres does it say there are in one metre.

A1: [re-reads briefing sheet] Ooo-ohhhhh! A hundred!

T: That’s right! So, if we know that 50cm is equal to 0.5m, do you think we can work out what 350cm is equal to?

A1: Three point fifty?

T: That’s close enough! We actually call that three point five! Now, returning to the problem in the question onscreen... Which do you think is the smallest distance?

A1: Either that one or that one [indicating to the two distances in metres, 4m and 4.1m]

T: Why?

A1: Well this one’s got a point one...

T: Hmmmm... Do you understand that Hundreds and Tens and Units are of different importance?

A1: Yeah.

T: Which is the most important of those three categories? Which is the biggest? Out of the hundreds, tens and units categories... which is the most significant?

A1: The hundreds?

T: That’s right! Now, think of decimals like this: Even though adding numbers after the decimal point makes the number appear longer, those digits have less significance. Think about money. Now, if I have £1 in my pocket... and I want to buy a milkshake, and that’ll cost me 99p. Now, .99 sounds like a lot, because 99 is quite a large number, but it’s not as much as one pound, because although one is a much smaller number than 99, the 99 comes after a decimal point!

A1: So, sort of like, decimals are less valuable than units....?

T: That’s exactly right! Wow! You’re a bright spark! Now! Let’s see if you can answer the problem. Which is the smallest distance on the screen?

A1: Four metres?

T: That’s right! So four metres is less than four hundred and forty centimetres, because centimetres are a smaller unit of measurement! On to the next question...
28> **A1**: [The participant clicks past the feedback screen. The fourth question appears onscreen.]

29> **T**: Aha! This is another centimetres conversion question…. Now, having looked at how we translate from centimetres into metres… Do you think you’ll be able to answer this question - have a quick go and see what you think… Which do you think is the smallest?

30> **A1**: 6.6?

31> **T**: 6.6m? That’s right! Well done!

32> **A1**: [The participant clicks past the feedback screen. The fifth question appears onscreen, featuring one sign reading 940cm.]

33> **T**: Now, remember what we said about centimetres being smaller than metres… Do you know how big 940cm is in metres?

34> **A1**: Rather big!

35> **T**: *Chuckles* yes, I suppose so! But in changing the units to metres, it seems less impressive…

36> **A1**: Oh. Nine point four metres?

37> **T**: That’s right, but to be very much exact, if we want to be very precise and clever, then we move the decimal point. Now, this is the quick and easy way to convert from centimetres to metres. So, look at these next three…

38> **A1**: Yeah.

39> **T**: Now, this is another tricky one… So putting a decimal point **there** would give us the metres…. [Signalling with pen over the screen.] So, which is the smallest?

40> **A1**: That one!

41> **T**: That’s right! And how about the next three? Which distance is smallest here?

42> **A1**: 8.4?

43> **T**: Yes, that’s right again! Not many to go now! How about these?

44> **A1**: That one!

45> **T**: That’s right And this time?

46> **A1**: That one!

47> **T**: That’s very good!

48> **A1**: That one!

49> **T**: Right! Well, didn’t you do well!

---

**Transcript of Participant B1**
The plot of the game is explained using the briefing sheet (Appendix A). Attention then turns to each of the images (Appendix B)]

1> T: Now, what I have here is a paper version of the software that I tested upon some of your classmates yesterday... See if you can identify the shortest distance from the road signs listed here. [Problem 1 is presented, the three distances are all in metres]

2> B1: [points to the correct answer].

3> T: That's right! Now, the questions get a little bit harder... [Problem 2 is presented]

4> B1: Not sure.

5> T: Okay! Well, let's refer back to this help sheet here. Can you see the differences between metres and centimetres? There are 100cm in every metre, and 50cm in half – or 0.5 - metres.

6> B1: Ah, ok [thinks] so it’s... that one?

7> T: You’re spot on! Right! I think that the rest of these might well be a little too easy for you! But let’s see how you get along. [Problem 3 is presented]

8> B1: Hmmm. That one? [indicates the right answer]

9> T: Now, why do you think that that number is smaller than this one [indicates to 7.4m and 834m respectively]... You’ve got the answer right, but I want to know how you know that 834cm is smaller than 7.4m...

10> B1: Well, centimetres are smaller than metres, and that’s going to be about 8m [refers to 834cm].

11> T: That’s right! Now let’s look at the next one. [Problem 4 is presented..] Oh! They’re all in metres. You should find this one easy!

12> B1: [promptly gestures towards the right answer] That one.

13> T: That’s right! Well, let’s see how well you get on with the next one. [Problem 5 is presented.] Tell me what that distance in centimetres translates to in metres before you tell me what you think the answer is.

14> B1: Hmmm.. that’s equal to... 2.66m.

15> T: Right! So, which is the smallest?

16> B1: Ah, ok [thinks] so it’s... that one?

17> T: “5.2m is equal to... 520cm. So remembering that, what do you think is the answer to [indicates question].”

18> T: Right! And the next one? [Problem 6 is presented.]

19> B1: That one.

20> T: Right again! And the next one? [Problem 7 is presented]
T: Right! Not many to go now... \([\text{Problem 8 is presented}]\)

T: That's right too! These are too easy for you aren't they? \([\text{Problem 9 is presented}]\)

T: Right! Okay, now the last one... \([\text{Problem 10 is presented}]\)

B1: Will this be on every computer?

T: How do you mean “every computer”?

B1: In the computer room?

T: No, not soon anyway. Why, did you like it?

B1: Yeah! it was funny. I like using computers. We’ve got four at home!

T: Gosh! You must have a big family!

B1: Well, they’re my Dad’s really, so I’m not allowed to play on them. Not on all of them.

T: Well, if you could help me by answering these questions, then I’ll be able to make the software even better, and then perhaps in the future I’ll be able to produce software good enough to give to your school! What do you think about that?

B1: Ok!
D Assessment Paper

The Adventures of Sir Eric the Unready

Which are ye? Bold Squire Fair Maiden

1. Change these lengths from centimetres into metres
   a. 300cm b. 800cm c. 125cm d. 765cm

2. Change these lengths from metres into centimetres
   a. 2m b. 10m c. 2.5m d. 3.6m

For the remaining questions, please write out the distances on the road signs in order, from smallest to largest

3. _____, _____, _____
4.

1977m 1797m
1828m

5.

3.9m 4.2m
406cm

6.
7. _____, _____, _____

8. _____, _____, _____
9.

Thank you for your help! 😊
### E Assessment Results

The results of the twelve participants who completed the assessment in Appendix D are recorded here. Questions 1 and 2 carried 4 marks, and subsequent questions carried one mark.

#### A1: Software Prototype

<table>
<thead>
<tr>
<th>Participant</th>
<th>Gender</th>
<th>Question Number</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Female</td>
<td>4 2 1 1 1 1 1 1 1 1</td>
<td>14 88</td>
</tr>
<tr>
<td>A2</td>
<td>Male</td>
<td>4 2 1 1 1 0 1 1 1 1</td>
<td>13 81</td>
</tr>
<tr>
<td>A3</td>
<td>Male</td>
<td>2 2 1 1 1 1 1 1 1 1</td>
<td>12 69</td>
</tr>
<tr>
<td>A4</td>
<td>Female</td>
<td>0 0 1 1 0 0 0 0 0 0</td>
<td>2 13</td>
</tr>
<tr>
<td>A5</td>
<td>Male</td>
<td>2 3 1 1 0 0 0 0 0 0</td>
<td>7 44</td>
</tr>
<tr>
<td>A6</td>
<td>Male</td>
<td>4 1 1 1 0 0 0 0 0 0</td>
<td>7 44</td>
</tr>
</tbody>
</table>

#### B1: Paper Prototype

<table>
<thead>
<tr>
<th>Participant</th>
<th>Gender</th>
<th>Question Number</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>Male</td>
<td>4 4 1 1 1 1 1 1 1 1</td>
<td>16 100</td>
</tr>
<tr>
<td>B2</td>
<td>Female</td>
<td>2 0 1 1 1 1 1 1 1 1</td>
<td>10 63</td>
</tr>
<tr>
<td>B3</td>
<td>Male</td>
<td>0 0 1 1 1 1 1 1 1 1</td>
<td>8 50</td>
</tr>
<tr>
<td>B4</td>
<td>Female</td>
<td>0 0 1 1 1 0 1 1 0 0</td>
<td>5 31</td>
</tr>
<tr>
<td>B5</td>
<td>Male</td>
<td>2 1 1 1 1 1 1 1 1 1</td>
<td>11 69</td>
</tr>
<tr>
<td>A6</td>
<td>Male</td>
<td>2 0 1 1 0 0 0 0 0 0</td>
<td>4 25</td>
</tr>
</tbody>
</table>
F Fun Toolkit Questionnaire

Presented here is a customised questionnaire incorporating the suggestions presented by Read, MacFarlane & Casey (2002) in their "Fun Toolkit". Their toolkit is designed to encourage feedback from children over the usability of an interface. In addition to this, children were encouraged to comment on this feedback.

SMILEY-O-METER

The smiley-o-meter is like a thermometer, but instead of measuring the temperature, it measures smiles!

Please help me to improve this software by letting me know a little about what you thought of the software.

**Graphics:** What did you think of the pictures used in the game?

- Awful
- Not very good
- Good
- Really good
- Brilliant

**Puzzles:** How did you find the puzzles featured in the game?

- Awful
- Not very good
- Good
- Really good
- Brilliant

**Ease of Use:** Was the game easy to play, or too complicated?

- Awful
- Not very good
- Good
- Really good
- Brilliant