A Logic Programming Tutor

This project aims to produce a software product capable of explaining undergraduate students how logic programs and their respective answer solver work.

Matthew James Scott

BSc (Hons) in Computer Information Systems 2005

Abstract

This project seeks to address the problem of producing a virtual learning environment for undergraduate students to learn logic programming with answer set semantics. The tutorial covers a relatively recent field in which the academic literature relies on a depth of background knowledge. An introductory tutorial has until now not been produced and hence this will assist the development of the discipline. This is a software development project and as such approached the problem using the waterfall model for software development. The resulting system allows students to read core subject content, ask tutors questions and perform self evaluations. The system also allows tutors to submit additional tutorials and evaluations. If up taken by the answer set programming community the range of tutorials will grow and the site will act as a first step into understanding logic programming with answer sets.

Acknowledgements

The author would like to thank Simon Wilison and Natalie Downe for their technical and moral support respectively. Special thanks go to Marina de Vos for her patience and virtue.

Signed Declaration

Copyright

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor 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 institute of learning. Except where specifically acknowledged, it is the work of the author.

Signed:

This thesis may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed:

Contents

1. Introduction

Over the past fifty years the field of logic programming has developed from academic discussion to an applied science of increasing relevance to undergraduate students. In the 1960's the theories of predicate logic were first applied in PROLOG, a LOGic PROgramming language. The next thirty years saw a realisation of both the power and the limitations of this language.

PROLOG was the first mainstream declarative programming language. This meant that the computer executed the program not based on a set of linear instructions but by an understanding of a set of facts and rules and by the computation of said rules by a logic engine.

Applications of PROLOG included data representation and simple diagnostic tools. However academia did not hesitate to identify the weaknesses of PROLOG's logic engine. The language was inflexible as the facts and rules needed to be defined prior to execution. There were issues surrounding it's understanding of the distinction between negation and negation as failure. Hence, the consequences of not being able to distinguish between a component failing and not knowing whether it has failed could be critical. It could not comprehend the fact that a rule could typically hold, such as birds typically fly with the exception of penguins and those that are dead. The logic programming community spent the next twenty years addressing these problems through the development of theorems such as default logic and Auto-epistemic Logic.

In 1999 was the defining year for answer set programming (ASP): - A declarative theory with its roots in logic programming and deductive databases. ASP has the possibility of becoming a truly effective knowledge representation tool(Christian Anger, Kathrin Konczak et al. 2002). Though syntax ally it looks very similar to Prolog, its logic engine works very differently. Answers set programming models a computational problem then uses an answer set solver to select a solution to the problem from within a set of constraints. This means that it is ideally suited to solving combinational search problems in knowledge representation and reasoning. Other applications include planning tools, medical diagnosis and space exploration. This combined with the rapidly expanding body of research in the field; it becomes clear to see why ASP has breathed life into non-monotonic logic and logic programming and how the next five years will see answer set programming and logic programming creeping on to undergraduate syllabi in a variety of disciplines.

While conducting initial research into the field of Logic Programming it became apparent that the body of knowledge is not accessible to an undergraduate student without a substantial prerequisite understanding of logic or declarative programming. There are limited textbooks available, the materials held within journals are not aimed to introduce a student to the subject and the tutoring material that is held online usually supplements a lecture course therefore provides little in depth explanation of concepts or tools to assess your learning. This, however, is far from unusual. Education journals frequently comment that e-learning is typically unstructured and implements little or no pedagogical theory.

During the course of this project a solution of to this problem has been implemented. The project set out with the clear aim to produce a software product capable of explaining undergraduate students how logic programs and their respective answer solver work. It will be also of academic interest to explore the feasibility of implementing commonly accepted training methodologies to an interactive e-learning environment.

During the course of this project a proposal to produce an online piece of ASP tutoring software was posted online (Costantini 2004). Due to the non-collaborative nature of the final year dissertation there was no involvement between the two parties except the agreement to exchange work after the final submission. Apart from this proposal there is no evidence of previous implementations of this project.

Prior to initiating the software development cycle a literature review was conducted focussing on the course content, learning theory and practical aspects involved in the development.

2. Literature Survey

2.1 The e-learning agenda in Higher Education

The wide spread use of the internet paired with the government agenda for widening access to Higher Education presents the university sector with the opportunity to radically alter its strategy for learning and teaching. “In the White Paper The future of higher education, the Government set out a vision for a higher education system characterised by inclusion, excellence, flexibility and collaboration. In meeting this vision, e-learning has an ever-more important role to play.” (DfES 2004) If the government wishes to attain it target of 50% participation in higher education then universities must alter their teaching methods to allow:

- Distance learning
- Part time courses
- Flexible pace of study
- Flexible points of entry

E-learning is arguably the pedagogical model that best fits these requirements. The internet allows students to access resources from various remote locations. Communication can be asynchronous and this allows flexibility in the pace the course is completed and entry points to the course.

It is a widespread misconception that e-learning is a relatively new methodology. In some form electronic learning has been around for over 50 years. In 1951 a two-way radio was used in Alice Springs School to facilitate distance learning. During the 1960’s the University of Wisconsin was using telephone conferencing to supplement paper based distance learning and the 1970’s saw the establishment of the Open University.

There is wide spread debate of whether this represents a solution or a new set of problems. There have been six potential problems identified with e-learning (S Ryan, B Scott et al. 2000). These include; high set up costs, extra costs of maintaining, revising and updating courses, the need for students to be motivated and self organised learners, the lack of peer contact and the interaction for students working alone, the need for flexibly available tutor support and problems with ensuring materials are pedagogically high quality.

This said; universities across the country are in the process of devising e-learning strategies to address these problems. The University of Bath has created a central e-learning team. “The University's Corporate Plan articulates the aim of establishing the University at the forefront of innovation in flexible/open learning … delivered via a variety of modes, media and methods, including e-learning.” (University of Bath 2002) This has been delegated to the e-learning team. There are already various forms of Virtual Learning Environments within the University such as Blackboard where resources such as lecture note can be placed and interaction can take place in the form of discussion boards. Though these are used in varying degrees between departments, the actions of Bath University are

replicated across the country and whether e-learning will simply support traditional forms of lecturing or eventually replace it is yet to be seen.

2.2 Logic Programming

Prior to exploring how e-learning can be used to facilitate teaching Logic Programming, it is necessary to investigate the actual content of the course. Logic programming can be defined as a programming paradigm in which the attributes a solution should have are specified as opposed to the set of steps that need to be taken in order to get to the solution. This means that the syntax of a logic programming language is distinctly different from that of a functional or procedural language. Logic programming is strongly based on formal logic where a concept is represented through a set of propositions, which are either true or false, the relationships between these propositions and new propositions that can be inferred from the existing propositions. This representation is known as symbolic logic of which predicate calculus is a subset.

2.2.1 Predicate Calculus

In predicate calculus propositions can be either compound or atomic. Atomic propositions consist of a set of compound terms which in turn consist of a functor and a set of parameters, i.e.

```
Father(David, Jaweh)                    (1)
```

Compound propositions are just sets of these atomic propositions connected by operators (see figure 1.1)

Variables can be introduced using quantifiers and conventionally start with a capital letter. There are two kinds of quantifier; the universal (equation 2) where for every value the variable can have the proposition after it remains true; and the

Operator	*Symbol*	*Example*	*Meaning*
Equivalence	≡	a ≡ b	a is equivalent to b
Negation	¬	¬a	not a
Conjunction	∩	a ∩ b	a and b
Disjunction	∪	a ∪ b	a or b
Implies	⊃	a ⊃ b	a implies b
Implies	⊂	a ⊂ b	b implies a

(Figure 1.1)

existential quantifier (equation 3) where the proposition will be true for some value of the variable.

$$\text{Universal Quantifier - } \forall XP \tag{2}$$

$$\text{Existential Quantifier - } \exists XP \tag{3}$$

2.2.2 Clausal Form

For use in computer science the predicate calculus grammar is too complicated as it allows two statements that mean the same thing to be stated in different ways. This is called ambiguity. The clausal form provides a template for propositions to negate this problem. A clausal term is of the form:

$$B_{n+1} \cup \ldots \cup B_m \supset A_0 \cap \ldots \cap A_n \tag{4}$$

The means if all of the A propositions are true then this implies one or more of the B propositions is true. The left hand side of the inference symbol is called the consequent and the right is called the antecedent.

2.2.3 Robinson Resolution

Robinson (1965) noted that there are some interesting properties with regard to inference with propositions. The concept is as follows. Suppose you have two propositions:

$$P_1 \subset P_2 \tag{5}$$

$$Q_1 \subset Q_2$$

$$\text{if}(P_2 \equiv Q_1)$$

$$\text{then } P_1 \subset Q_2$$

The implication of this is that two propositions can have their antecedent and the consequent terms combine and any repetition of terms on either side of the resulting proposition are removed i.e.

$$f(x, y) \cup g(z) \subset h(x) \tag{6}$$

$$h(y) \subset f(x,y) \cap f(y, z)$$

$$\text{Combine:}$$

$$h(y) \cup f(x, y) \cup g(z) \subset h(x) \cap f(x,y) \cap f(y, z)$$

$$\text{Reduce:}$$

$$h(y) \cup g(z) \subset h(x) \cap f(y, z)$$

2.2.4 Proof by Contradiction

An important aspect of resolution is the ability to detect inconsistencies in theorems. If you insert the negation of the theorem as a proposition within the

theorem then resolution can reduce the theorem and find an inconsistency. The original proposition is called the hypothesis and the negation is called the goal.

In theory this is a sound practice however with large numbers of propositions within a theory it can take along time to process. Horn Clauses (Horn, 1951) are used to reduce the processing time. These are shorter versions of clausal form and can be of two forms. Firstly where there is a single proposition in the head (i.e. n = 0). This is called a headed Horn Clause. Secondly where the consequent is empty this is called a fact.

2.2.5 A history of logic programming

Though it is possible to trace the roots of logic programming right back to ancient Greek philosophy the first developments that really separated the field from logic happened around the turn of the century. Frege developed what we now know as predicate calculus in 1879 on the back of DeMorgen's and Boole's work on propositional logic. Once Herbrand and Godel proved that the proof machinery of predicate calculus could provide a formal proof for every logically true proposition the course was set for the development of automated theorem proving.

Alan Turing made significant contributions to the field; firstly, by demonstrating the Negative property of predicate calculus (it is not possible to produce an algorithm that determines that something is not a logical consequence of something else); and secondly through the war work that lead to the development of the Universal Turing Machine. By the 50's the hardware was developed enough to support computational experiments with predicate calculus. This lead to a move from human orientated languages to machine orientated languages.

In the run up to the seventies there were heated debates between Edinburgh and MIT AI groups as to whether it was better to represent knowledge in a procedural or declarative way. 1972 was the advent of Prolog, developed by Alain Colmer and Phillippe Roussel at the University of Aix – Marseille with some assistance of Robert Kowalski at the University of Edinburgh. Prolog was the first logic programming language to be widely adopted and used for commercial applications. The community was aware of the limitations of Prolog and continued to propose implementations for non-monotonic logics such as default logic. Unfortunately these implementations were slow and often complex. Hence, Prolog was logic programming language of the latter half of the 20th century.

2.2.6 Prolog

Prolog Syntax

Atoms: A string of letters digits and underscores that begins with a lower case character.

Constant: Either an atom or an integer.

Variable: A string of letters digits and underscores that begins with an upper case character. They are not bound to types by declaration. This happens in the resolution process.

Structure:	Much the same as an atomic proposition in predicate calculus. These consist of a functor and a set of parameters, (see equation 1). The functor is an atom and used to identify the structure. The parameter list is any set of other structures variables and atoms.
Predicate:	A structure whose atom is a question (a goal statement).
Terms:	A term is a variable, constant or structure.
Fact:	A headless horn clause
Rule:	A headed horn clause. In Prolog conjunction is implied by a comma (,) and inference by (: -). The end of a fact or rule is denoted by a full stop.
Goal:	This is when the system is queried by using a predicate. If the predicate uses atoms or constants then the system will return yes or no. When it has a variable in the parameter it will find values of the variable where the goal is true.

How Prolog Works

Once a goal is input by a user the inference engine must find a chain of facts and rules that connect the goal to one or more facts in the database. It takes a proposition and searches from the top of the list of facts and rules the user has defined to find one that matches. This is called pattern matching. The problem arises when the right hand side of rules have multiple facts and or variables in them.

There are two ways in which Prolog inference engines make these chains. The first way is known as forward chaining, where the engine starts with the facts and rules and tries to find a sequence that leads to the goal. The other way is backward chaining where the engine works from the goal to the facts and rules.

Systems of fact and rules could often create lots of chains heading from the facts to the goal. The question that the people who make the inference engine have to decide is whether they take each chain at a time and find out if it goes all the way between the facts and goals or whether they deal with lots of chains at once and take them all one step at a time. The engines that deal with one chain at a time are called depth first engines and the engines that deal with lots of chains at once are called breath first engines. Prolog's designers decided to choose to deal with one chain at a time (depth first) as it was less processor demanding.

The last thing you need to know about the inference engine is that when it is looking to see if a chain of facts and rules match a goal and it realises that the two don't meet then it backtracks to a fact or rule where it has already been and sees if it can join up in another direction.

Problems with Prolog

Closed World Assumption

Prolog assumes that all the relationships that exist between the constants specified in its list of facts and rules are stated within that list. If there is something that is

true on the outside world and Prolog doesn't have enough information to prove it true then it will tell you it is false. It can't say, "I don't know."

Resolution Order Control

Prolog allows the user to determine the order of pattern matching. This is due to its depth first system of inference hence the first goals and facts that are listed will be matched first. The problem with this is, though it can be used to make programmes more efficient, it is easy for a programmer to program an infinite loop.

For further efficiency Prolog uses the non-logical operator cut (represented by the '!' symbol). This allows the user further control this time over the process of backtracking. This states that if certain goals fail then the inference engine does not try to satisfy the sub goals that came before them (on the chain that is being backtracked) by matching them to other routes to the final goal.

Negation Problem

This problem is seeded in the fundamental logical set up of Prolog. Prolog uses Horn Clauses. Horn clauses us positive logic. For example if all of the antecedent proposals are true then the consequent proposal is true. However there is no way of inferring from the state of the antecedent proposals to what degree the consequent is false.

2.2.7 AnsProlog*

The aforementioned problems with Prolog mean that it is not entirely suitable for the task of representing large amounts of knowledge. For some time the knowledge representation community have been attempting to find a new tool that negates these problems. Chitta Barrel (2003) proposes AnsProlog*, or answer set programming language, as this alternative tool. AnsProlog deals with the Prolog limitations, as:-

- It allows multiple terms in the consequent (head of rules) this means that a range of possible solutions can be proposed (an answer set).
- It does not use non-logical operators such as cut.
- It has two forms of not so that negation as failure is not an issue.
- The answer set semantics do not define a query processing methodology so the implantation of the engine can be chosen to suit the knowledge base.

AnsProlog Semantics*

The AnsProlog* programs has two languages; a language for storing the facts and rules (axiom language) and a language for knowledge from those facts and rules (query language). Each of these has an alphabet which is used to construct the axioms or queries. The axiom alphabet is defined as follows:

Object Constant: These are strings of letters and digits beginning with a lowercase letter that represent an entity within the domain of

	knowledge that is being represented. This could be a heart in the domain of a medical expert system or a task in the case of a planning system. When dealing with abstract constants they are often referred to as a, b and c.
Variable:	A string of letters digits and underscores that begins with an upper case character. When dealing with abstractions it's conventional to use X, Y and Z symbols.
Functions:	These are of the form who_is(the_king) and return an object constant using the values of the parameters within the bracket. Conventionally f, g, and h are used to represent functions.
Predicate:	These are used to describe the properties of the current state of the domain that is being represented. For example on(jug, table). p and q conventionally represent abstractions of predicates.
Connectives:	These include ¬, or, ←, not, ',' exclusively and are used to construct more complicated predicates.
Punctuation:	These clarify the expression of relation and are limited to '(', ')', '.'
Special Symbol:	This is ⊥ an represents and empty headed clause.
Term:	Either an object constant (referred to as constants) or a variable or $f(t_0, \dots, t_n)$ where f is a function and $t_0, \dots, t_n$ are terms.
Ground Term:	A term with no variables contained within it.
Herbrand Universe:	(HU_L) The set of all ground terms contained within the specified language.
Atom:	$p(t_0, \dots, t_n)$ where p is a predicate and t_i is a term
Ground Atom:	An atom with only ground terms within it
Herbrand Base:	(HB_L) The set of all ground atoms within a specified language (L)
Literal:	An atom (possibly preceded by ¬)
Positive Literal:	An atom
Negative Literal:	An atom preceded by ¬
Naf Literal:	An atom (possibly preceded by the symbol 'not') can be negative (latter) or positive
Gen Literal:	A literal (possibly preceded by the symbol 'not')

From these Gen Literals a set of rules are constructed in the form:

$$L_0 \text{ or } \dots \text{ or } L_k \leftarrow L_{k+1}, \dots, L_m, \text{ not } L_{m+1}, \dots, \text{ not } L_n. \quad (7)$$

The literals to the right of the '←' symbol are called the head and those to the left are called the premise or the body. There are various special types of rules. A fact is a rule with a single literal in the head and no body is a fact. A rule with an empty head is a constraint an a rule with literals in both the head and body is known as an implication..

A grounding of a rule is a set of rules that contains all the possible substitutions from the Herbrand universe to the values of variables in the rule. Hence, an answer set language is defined by the grounding of all the rules within the alphabet.

*Applications of AnsProlog**

RDMS:	When relational databases are queried the engines need to find chains of identifiers between the entities. The inference engines in Logic Programming are designed to do this.
Expert systems:	Expert systems store huge amounts of facts and rules of how to infer facts from other facts. Logic programming makes it easy to store and query these large databases effectively.
Natural Language Processing:	This was the original intent of Prolog. As words are grouped into noun phrases (the book) and verb phrases (to duck) logic programming languages rules and terms can infer that a certain combination of words can grammatically go together.

Other Applications include; planning tools, knowledge representation and product configuration.

Part of the AnsProlog appeal is that it can restrict the use of symbols to create syntactic sub classes such as AnsDataLog where the language does not use function symbols. Each of these syntactic subclasses exhibit different levels of expressiveness and complexity so we can use different classes for different applications and hence, the language is flexible. It does not allow arbitrary formulas to be used and so it seems much less intimidating but is just as expresses. Unlike other languages which allow the addition of rules to alter the existing answer set AnsProlog* has various efficient implementations currently in use. Barrel (2003), among others, believes that logic programming has "a lot going for it" to be the leading language for knowledge representation, reasoning and declarative programming. AnsProlog* has a growing base of support and as a language is building the momentum that will push it onto the undergraduate syllabus in a variety of disciplines. The challenge now is how the theories, concepts and facts around logic programming with answer sets can be delivered accessibly to the undergraduate mass.

2.3 Learning Theories

Various models have been proposed for the way in which we learn. One of the most commonly acknowledged models is that of which David Kolb (1984) proposed: the experimental learning cycle, see Figure (a). The theory that we learn from experience can be traced back to when Confucius (450 BC) stated:

"Tell me, and I will forget. Show me, and I may remember. Involve me, and I will understand."

Kolb's model works from this principle and states we learn in a cycle of four stages. Initially we experience an outcome from a set of events, be it a theory instructed in a lesson or a practical experience like burning a cake in an oven. The mind then records this information for later use. This information is formed into a hypothesis that if these set of events occur again then the following outcome

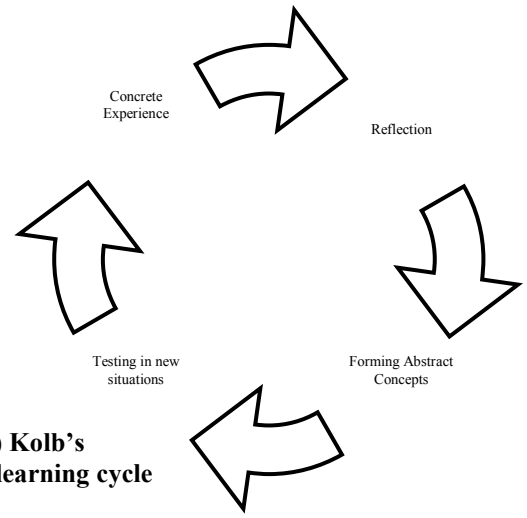


Fig (1.2) Kolb's experimental learning cycle

should occur once more. Each time this set of events or theory is applied to a new situation it is either reinforced or modified in light of new information.

Kolb's goes on to identify a typology of learning styles and states that these different learning styles affect which stage in the process the learner will perform most effectively. It is important that we accommodate different learning styles within the design. There is a multitude of classifications of learner styles, from Honey's (1982) Activist – Pragmatist – Reflector - Theorist to Kolb's Converger - Diverger – Assimilator - Accommodator. It is important that the a tutorial accommodates for different learning styles by providing a range of activities and uses of media.

The major criticisms of the model revolve around three lines of argument (Jeffs and Smith 1996). It is believed that in reality the steps maybe happening concurrently. The model does not take into account of cultural conditions of the learner and the model is not supported by scientific observation.

The abstract arguments of the model seem to be intuitive and a whether the stages are happening concurrently or as a process is not relevant to the process of designing a tutorial. This is because if we acknowledge that the stages in the process happen then it is more important that the tutorial is designed to include

reiteration and theory by example than it is than each stage of the tutorial follows this learning cycle.

Arguable the critique that is the most applicable to the Logic Programming Tutor is the fact that the cycle may not be relevant to a broad cultural spectrum. Learners from a culture which has a large power distance may not feel free to question authorities that deliver the course and this could be exacerbated by the isolation that is inherent with non face to face learning. The model also fails to account for group learning. In communal cultures learners are more reliant on consensus opinion than their own and also may not feel free to question given assumptions.

2.3.1 Teaching Methods

Prior to designing a course it is prudent to examine various teaching methods used in didactic teaching techniques. The Government's 'National Numeracy Strategy' identifies three core elements for teaching mathematics in the classroom. These are; direct instruction, interaction and assessment.

Direct Instruction

This can be defined as generic core knowledge transfer (Muijs and Reynolds 2001). This core knowledge represents the methodology, theory, issues, factual content and specialised vocabulary and concepts that need to be communicated to the learners in order that the learning outcomes are attained. The main elements of effective direct instruction include; clearly structured lessons, correct pacing, intuitive modelling of concepts, the use of concept mapping and reinforcement of this using interaction.

The structure of lessons is of paramount importance. A well used technique is to recap the lessons work at the end of the lesson and start of the next lesson. The students need to justify why they are partaking in the lesson so it is useful to outline the learning outcomes of the lesson at the start of the lesson and make them available outside the lesson. This can be also used as a framework to recap the lesson at the end. Effective teachers use intonation, multimedia, repetition and redundancy to highlight key points within the lesson.

The actual content of the lesson should be structured into a number of small clear sections or topics. These should also use the aforementioned structuring techniques; however they can use further methods of presentation to enhance clarity. A subsection can be presented in a deductive manner (working towards a conclusion from abstract concepts) or and inductive manner (working from real life examples to general rules of principles). Other structures include the part-whole format and sequential ordering (Borich 1996). The part-whole format involves teaching a concept in its general form the teaching specific parts or examples that explicitly relate to the whole. For example in a programming concepts lecture presenting the conditional concept the introducing pre-conditioning and post conditioning. In sequential ordering topic are taught in the order they occur within a process for example the process of making bread.

The pace of the lesson is very much dependant upon its content however as a general rule a fast pace of learning will aid momentum, keep interest and

efficiently cover the course content. However pace must not come at the expense of clarity and understanding. An effective teacher continuously assesses the learners' understanding allowing time for feedback and reducing pace over more complex topics. Once the core content has been delivered through these small steps an effective teacher will assess the classes understanding of the content and allocate further reading or work to reiterate this understanding.

Interaction

Interaction takes place between teachers and learners and between individual learners in order to check progress, restate their understanding of a topic, increase self esteem and to support the topic with other examples (scaffolding). Interaction in the form of questioning the class can be of two forms, open and closed. Closed questions require a straight forward answer and students according to Reynolds (2001) should be given around 3 seconds to answer. Open questions require more thought however as they could be explaining a process or justifying an opinion. In a class room situation Reynolds states that 15 seconds is adequate whereas Exley (2004) states that this can cause awkward silences and so group work can be used it alleviate this. Whether to use a majority of open or closed questions and the complexity of said questions is also up for debate. The general consensus is that effective teachers use more open ended complex questions than less effective teachers however they still use a majority of simple closed questions to ensure the class remains focused on the lesson.

Another aspect of interaction within a lesson is seatwork. This consists of exercises that review and practice the content covered through direct instruction. This also allows the teacher to move around the room and assess individual learner's understanding. Teachers need to produce enough exercises to occupy the range of abilities within the class and to clearly explain what needs to be performed and how this will help achieve the learning outcome. Learners need to be able to clarify the task and gain feedback from it. The feedback can be self assessment.

Assessment

There are various ways of classifying assessment. The most common is whether the assessment is formative or summative. Summative assessment leads to formal accreditation where as formative identifies the strengths and weaknesses of the learner, clarifies points and suggests methods of improvement. It is debatable whether feedback needs to be given of summative work. Closed question assessments are those which result in a set of results that can be easily distinguished as right or wrong. They can be used to assess a learner's grasp of factual content specialised vocabulary on concepts. They include multiple choice and cloze (fill in the blank) tests. They are quick to perform and mark so hence more of the syllabus can be covered. However, writing the initial questions takes longer to prepare and the results don't explain thinking and could be based on guesswork.

Open question assessments are those that are constructed to measure the learners understanding of theory, methods and issues and don't have a determined outcome. These take longer to mark as they need to have a very stringent marking

scheme that states how learners can demonstrate the learning outcomes they have achieved. The markers need to be skilled enough to distinguish between essay skills and understanding of the course. Blind marking is required for open tests as the marking is left open for interpretation of ability. A well constructed open test gives an accurate portrayal of a learners understanding.

Practical skills can be measured through qualitative analysis of a set of course works and quantitative measurements of a performance of that skill. These are known as portfolio assessment and performance assessment respectively. These methods of assessment are known as aptitude tests as they measure a skill where as open and closed assessments measure understanding of the curriculum and so are known as achievement tests.

There are various measures of whether a particular test is a suitable means of assessment (Gronlund 1991). They must; be constructed to measure learning outcomes, encompass different types of knowledge, produce consistent average score and be able to identify student strengths and weaknesses in order to give meaningful feedback.

Independent Study

As learners mature so do their learning skills and hence they can study independently in a much more competent manner (Duggleby 2000). It is therefore wise to note how homework is used effectively in a class room environment.

There are three kinds of homework tasks; preparation for the next lesson, practicing new material and extension of knowledge. Homework should increase students' performance in achievement tests reinforce content covered and complete work unfinished in lesson time. It also has a role to play in skills development such as improving discipline, time management, research and study skills. According to research measurable positive affects of the use of homework include increased short term retention of knowledge and long term development of skills, while homework can also negatively impact on students' motivation and self esteem (Cooper, 1989).

Muijs (2001) identifies four key principles for setting work outside lesson time. These being:

- It should not be used as a punishment
- It should be used to provide diagnostic feedback
- It should be aligned to learning outcomes
- It should be integrated into activities carried out in a student's home life

2.3.2 Teaching mathematics

The generic core content of the Logic Programming Tutor will be quite mathematical due to there being a close relationship between mathematics and formal logic which is the basis for logic programming (Sebesta 1996). It is therefore follows that we should examine effective teaching of mathematics within the classroom.

From the point that learners enter education they may have misconceptions about mathematical concepts. For example children can to a large extent count before they come to school but their understanding of cardinality is shaky. In higher education misconceptions can be much more deeply ingrained. These misconceptions need to be addressed by teaching advance definitions from the point of inception.

Learners often have problems relating mathematical concepts to their understanding of the real world. This is could be due to the abstract nature of mathematics and can lead to lack of motivation in the lesson. Mathematical concepts need to be supported by real life worked examples and analogies as much as possible linking them back to other areas of knowledge (Muijs and Reynolds 2001).

2.3.3 E-Learning Delivery Styles

There are multiple methods of delivering e-learning opportunities to students. This is succinctly illustrated using the Online Paradigm Grid as shown in Figure a (Stephenson 2001).

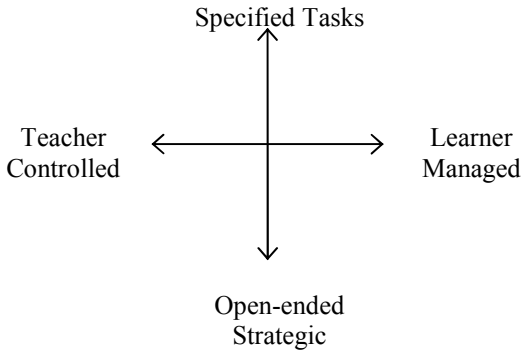


Fig (1.3) Online Paradigm grid

The grid divides e-learning into four paradigms. Teacher determined – task specific where the teacher specifies the points of interaction, online content and learning outcomes leaving little room for initiative. The learner determined – task specific paradigm gives the learners the freedom to control how they progress towards a specified set of learning outcomes. In the teacher determined – open ended strategic learning activities the course is initiated in a similar way to task specific paradigms however after the core course content is covers the learner is free to explore the information online in an unstructured manner. In the final paradigm, learner managed – open ended strategic learning activities, the learner engages in personal goal setting exercises and directs themselves through exercises facilitated by the teacher to attaining said goals.

This is all very well in theory however it is prudent to identify the practical implications of these concepts. The main implication is that of staffing resources. If course are to be highly staff lead then this will mean that there will be a larger demand of staff time. However if staff are to facilitate learner managed learning it could be foreseen that the tutor will have to manage multiple bespoke courses and this could be equally time demanding. If a course entails a set of narrowly

specified user tasks then the course set up costs will be higher as the institution will have to underwrite the software development costs of performing these tasks online. That said with a more open ended approach to course tasks the process of giving the learner feed back entails higher costs due to the task's non-alignment with learning outcomes. In the course of formulating requirements for the Logic Tutor Program it will be necessary to assess the resources available to deliver the course and find a happy medium within these paradigms.

2.3.4 Computer Mediated Communication

"Fundamentally university is a community holding conversations about knowledge," (Ryan 2001 quoting Daniel, 1998). Within the traditional, bricks and mortar, University these conversations about knowledge happen in various ways and hence it is interesting to compare them to examples of computer mediated communication:

University	*E-Learning*
Lecture	Web based materials / Conference
Tutorial	Chat Rooms
Group coursework	Shared Authoring Tools
Individual coursework	E-mail, Shared Authoring Tools
Closed assessment	Computer Aided Assessment
Open assessment	E-mail, Shared Authoring Tools
Peer Learning	Chat, E-mail, Forums, MUDs,
Open Query	Forums
Confidential Query	E-mail

(Figure 1.4)

Computer-Mediated-Communication defines the ways that telecommunication technologies have converged with computers and digital networks to create a new set of tools to support human communication (Ryan, Scott et al. 2000). Though this sound quite ambiguous it's the "new set of tools" that will be explored in this section. One of the key problems with distance learning before the advent of the internet was the perceived feeling of isolation by learners (Duggleby 2000). This was due to the lack of interaction that a traditional classroom environment provided. The "new tools" can be used to address the deficit in communication. Though many of the tools mentioned within the table are commonplace and need little discussion, it is necessary to elaborate on a few.

Web-based materials

The web uses a range of media in a non-linear way. This makes it a very different information source than that of the tutor in direct instruction. To simply put lecture note on the web would be a waste of a medium. Many online tutors, such as the Java Tutorial at the Sun-Microsystems site, lead the users through a series of text based trails to meet the learning objectives. When a user needs further explanations the point in the site is dynamically linked to examples which scaffold the learners understanding. However, studies have shown that it takes longer to read from a screen than from paper and people prefer to read printed text than that from a monitor (Nielson 1989). The use of multimedia can elevate dissonance in learner learning styles and hence increase motivation an interest (Nielson 1989; Given 1996). There are various ways to present this range of media. It can be presented in a sequential manner, much like a slide show or in a more dynamic manner allowing the user the freedom to browse through the material in there own way. A sequential manner is often preferred as it allows the learner to build upon their understanding from a ground level. This is called a constructivist approach to learning.

The World Wide Web consortium (W3C) has recently released a new mark up language known as SMIL. It allows the temporal behaviour of multimedia objects such as video and audio files to be described in a dynamic context. The advantage of this is that now there is a standard for representing synchronous material on the web.

Shared Authoring Tools

These allow learners to work on the same document at the same time. This can be as simple as a Wiki which is a webpage that allow any members to author the content to environments that permit multiple cursors on the screen at the same time. This supports collaborative learning and allows learners to observe how other learners work. There are some problems that designers need to consider. These include the fact that with large documents it can be hard to find what other people are referring to and that how the software deals with learners updating the same piece of information at the same time. Shared authoring is also useful for transparent assessment of work for open style questions as this allows the tutor to provided “red pen” feedback on the actual learner’s transcript.

Computer Aided Assessment

The process of assessment of closed questions is fairly simple. The marker collects the learners’ responses then compares them to the answers. This information can be used to identify and overall score and pin point areas of weakness. Computer Aided assessment is where this process is automated using a computer.

The one main draw back of this method of assessment is that it is virtually impossible to detect learner collaboration. Two methods have been proposed to deal with this; firstly, selecting a random subset of questions on the topic for each learner or alternatively you can devise an algorithm to detect similarities in the learner’s results.

The type of exercises that can be performed online are only really limited by the designers imagination. However the learners' answers must result is something that is recorded to a stored value. Suggested exercises include; multiple choice; fill in the gap; short answer; jumbled sentence; and crossword puzzles. Java script source code is available for these exercises at http://web.uvic.ca/hrd/halfbaked/index.htm.

Multi-User Dimensions

These are 3D / 2D virtual environments users explore and engage with other users. Predominantly text based, MUDs were originally designed for role playing games and during the mid-nineties educational practitioners were using them for course based interaction (University of London Birbeck College 2004). Interestingly enough Birbeck Colleges' MUD, BioMoo is soon to be taken off line. A MUD differs from a chat room as it is object oriented so users can create object within in then other users can react with these objects once the creator has logged out. Within the context of the Birbeck Biology department the tutor would schedule meeting and set agendas with papers for the students to read and then the students would log on at a given time. The disadvantage of this is that it is a form of synchronous communication and hence needs all members to be available.

2.4 Practical Aspects of Development

2.4.1 Designing the course content

The University of Bath already has various quality assurance documents to guide the inception of new courses. (University of Bath 2005) Though guidelines are given as a list of considerations that need to be made prior to creating a new course, they do not explicitly state how to go about designing one. Ryan (2000) proposes the following framework for resource based learning course design:

Initially a needs analysis is conducted where the number of potential students are identified along with the subject field and the over arching course aims. Duggleby (2002) also suggests that initial planning should include an initial feasibility study to identify the level of academic and technical support the institution can give this course and a search and analysis of existing practitioners in the subject's field.

Specifying learning outcomes is the next point of action. This involves writing down what skills knowledge and experience you expect the learners to leave the course with. These should be stated in a way that both the practitioner and learner can understand.

Duggleby (2000) and Ryan (2000) take different approaches as to when to identify the assessment procedures. Duggleby believes it should be don't immediately after specifying the learning outcomes whereas Ryan believes it should be done after designing the content, tutorial strategy and support systems. This really depends as to what level you specify your learning outcomes. Developing the course content before the assessment procedures will allow you to elaborate on precise factual detail within the course so to know what to assess. However this could lead to assessing what the learners were taught as opposed to what the course set out to achieve.

The next step, according to Ryan is to identify what exactly is and isn't in the course content. Useful techniques for doing this include spider diagrams and relevance trees (Dawson 1999). From these abstract diagrams of the course it is easy to break up the course into topics and then into what is to be covered in individual sessions.

Breaking the course into sessions means that the learner has time to reflect on what they have learnt and terminate there learning at relevant points. It also will allow the tutor program to assess their learning and recap key information.

The strategies to adopt within these sessions then need to be specified. This includes a range of media to suit different learning styles and variance between direct instruction and interaction. Each of these activities need to be allocated a duration.

Once you know the course content and the section plans you must identify what support services are required for the student to complete these activities. These might include interaction with the tutor and the other learners, the conferencing tools the system will provide or any study guides that will be provided.

The assessment procedure then needs to be closely aligned to the learning outcomes - correlating what needs to be assessed to activities that can be simulated by the tutor software.

The penultimate two steps are development and implementation. By development Ryan means the coding of the website, preparation of printed materials, training of department staff and the production of a plan of implementation. Implementation occurs when the software goes live. Finally after the first course is over Ryan suggest evaluation. This should be 360°. Feedback should be collected from academics, students and technical staff. This information should be reported and disseminated to the staff involve and then the process should start once again. Ryan leaves feedback right until the end of the process. This is also true for Duggleby. I believe that this may not lead to a particularly usable system.

2.4.2 Designing for Usability

Sommerville (2002)identified six interface design principles from a longer list given by Schniederman (1998) (see fig 1.5).

(Nielsen and Molich 1990) however, identified 10 heuristics for usability. Negating any repetition of principles, Neilson states that the design should focus on prevention of error as well as recovery. He believes in transparency of information so that the user should not have to remember information and there are clear instructions throughout. He believes that users should be able to use fast keys and tailor frequent interactions. He also advocates an aesthetic and minimalist design with no irrelevant data displayed.

It is important that we incorporate processes for designing for usability within the design to ensure that the design is user centred. The US government have produced a cite on usability called Research Based Web Design and Usability Guidelines which identifies four key processes in designing for usability these are:

Principle	Description
User Familiarity	The interface should be in terms and concepts which are drawn from the experience of the people who will make the most use of the system.
Principle	Description
Consistency	The interface should be consistent in that, wherever possible, comparable operations should be activated in the same way.
Minimal Surprise	Users should never be surprised by the behaviour of the system
Recoverability	Interfaces should have mechanisms to allow users to recover from errors
User Guidance	The interface should provide meaningful feedback when errors occur and provide context sensitive user help facilities
User Diversity	The interface should provide appropriate facilities for different types of system user

(Figure 1.5)

- Set and state the goals of the site - some sites even state there purpose on the front page of the site
- Set Performance goals for the site - such as length of time to retrieve a give statistic
- Share independent design ideas – this is about disseminating best practice outside development teams
- Create and evaluate prototypes – this means iteratively paper prototyping

Process two is widely supported by the GUI community (Whiteside, Bennett et al. 1988) however the assertion of usability metric relies very heavily on measuring specific actions and the only sound test of usability is observation of users and hence usability engineering must go hand in hand with iterative design and prototyping (Dix, Finlay et al. 1997).

There are various approaches to prototyping; throw-away, where the prototype is built tested and then discarded with the design knowledge acquired documented; incremental design, where a more modular approach is taken; and evolutionary development where the prototype remains as the basis for the next prototype. Prototypes can be developed using a variety of techniques. One such technique is story boarding where the user is shown snapshots of the proposed interface. This can quickly produce an initial prototype however the process sacrifices observing

user interaction. A solution to this is to use a technique called limited functionality simulation. This is where the user is presented with a user interface that connects to a null data structure that returns set values without processing information. There are numerous tools available for this such as hyper card and python card.

There are two main dangers that a designer needs to be aware of when involving users in the initial design process the first is to constantly review decisions and not to allow an initial user comment adversely affect the rest of the design of the system. Secondly the designer need to be aware not just the symptoms of the users problems but also the reason behind the problem. Sommerville (2002) identifies some problems with iterative prototyping. The most obvious is that the process of prototyping is timely and often managers don't understand the resource demands attached with it. Also, as increments should be small and each increment should deliver some functionality, it is difficult to map the customer requirements into increments of the right size.

2.4.3 The DDA and SENDA

The Disability Discrimination Act (DDA) 1995 was amended by the Special Educational Needs and Disability Act (SENDA) in 2002. It is now unlawful for institutions to treat a disabled person 'less favourably' than they treat, or would treat non-disabled people for a reason which relates to the person's disability. For example, it would be unlawful for an institution to turn a disabled person away from a course, or mark them down in an assessment because they had dyslexia or were deaf. This includes people with; physical or mobility impairments, visual and hearing impairments, dyslexia and dyspraxia, medical conditions and mental health problems. The University can also be charged for failing to make reasonable adjustments within the University to accommodate non able bodied persons. The Disabled Rights Commission (2003) makes the following recommendations regarding web usability;

- Reduce the number of links and ensure that genuine and necessary links are clearly identified as such;
- Avoid site fragmentation: navigation mechanisms should be consistent (eg in appearance and behaviour), the relative importance of different sections (across the site and within pages) should be apparent, mark-up languages should be used to indicate the structure of pages;
- Preserve links to the Home page;
- Improve search design;
- Eradicate excessively deep site structures; and ensure that page titles are informative.
- In addition, the Guidelines should place special emphasis, in the form of elevated prioritisation, on the following matters already covered;
- The need to divide blocks of information into more manageable units;
- The need to ensure that foreground and background colours have sufficient contrast;

- The need to provide a text equivalent for every non-text element;
- The need to avoid creating pop-ups and new windows without informing the user;
- The need clearly to identify the target of each link;
- The need to use the clearest and simplest language appropriate for the site's content;
- The need to ensure that pages work when scripts and applets are not supported;
- The need to avoid movement in pages until they can be frozen.

The Quality Assurance Agency, a regulatory body for quality at University, gives the following code of practice for program delivery (cited from Exley, 2004):

- "The delivery of programs should take into account the needs of disabled people, or, where appropriate, be adapted to accommodate their individual requirements. Institutions should consider making requirements that ensure all academic and technical staff;
- Plan and employ teaching and learning strategies that make delivery of the programme as inclusive as possible;
- Know and understand the learning implications of any disabilities of their student whom they teach and are responsive to any feedback;
- Make individual adaptations to delivery that are appropriate to for particular students.

2.4.4 Interface Evaluation

Preece (Preece, Rogers et al. 2002; Disabled Rights Commission 2003) identifies various techniques for analysing a user interface. The first method, "Quick and Dirty," involves getting feedback from interviewing users. This is relatively non-demanding and may prove useful in design. Secondly is usability testing this is task orientated and all user inputs are recorded. Field studies test the user in a natural setting this testing is more relevant in safety critical systems and really relies on a judgement call on the testers' behalf of whether the natural environment is significantly different to the testing environment. The final paradigm identified is predictive evaluation. This is where experts use a heuristic framework to identify weaknesses in the design. Caution must be taken to ensure that these heuristics are backed up by concrete empirical evidence.

Prior to choosing an evaluation paradigm it is important to assess the factors effecting your decision (Dix, Finlay et al. 1997). The stage in the design / implementation process is important as to the level of functionality you can test. As mentioned before, there are trade offs between whether you test in a lab or in the field. You also need to consider whether the participants in the test will be objectives in their roles. It is often difficult to detect errors in your own code this can also apply with user interfaces. You need to determine what exactly you wish to find out and how this information is best retrieved. The presence of the

evaluator can sometimes affect the results of the test and so it needs to be decided whether their presence is beneficial. Also the urgency of the response also needs to be considered. A piece of software could be sent to and office a week before and interview is conducted and information may get forgotten in the process. The final consideration is the bottom line- What are the financial, staffing and time resources that are available to conduct the analysis.

Course Effectiveness

(Duggleby 2000) suggests a set of qualitative measures that the learners can rate upon completing an e-learning course. These include:

- Quality of Pre Course Guidance
- Quality of Web-Based Materials
- Quality of Tuition
- Course Workload and Pacing
- Quality of Assignments
- Quality of Group Work
- Technical Issues
- Value for Money

These are purely qualitative measures of course effectiveness. This makes it hard to rate the success of the course against similar courses delivered both through distance learning and the lecture hall. There is a dearth of any hard and fast guidelines on the subject in the UK. The Australian Flexible Learning Framework, among others, suggests various standards that an online course can aim to meet. Other academics (Sonwalkar 2002), have devised tools to calculate a rating of pedagogical effectiveness of an online course. A high rating is gained by using a range of media accommodating a range of learning styles and using a range of methods of interaction. With a lack of guidance from UK quality bodies such as the Quality Assurance Agency, it may be necessary to devise a bespoke method of evaluation with a strong emphasis one user acceptance testing.

2.5 Summary

The literature review has demonstrated aspects of the course content and e-learning that will be used in the design of the logic tutor. Further research needs to be conducted as to ascertain the style of delivery for the course and methods of evaluation that will be used for assessing the interface and overall course design.

3. Requirements Analysis and Requirements Specification

3.1 Requirements Capture Process

This section documents the way in which the functions and properties of the tutorial were ascertained. The functions and properties of the sight were gathered in slightly different ways so it makes sense to address them separately. After this there will be a brief discussion about the process by which the requirements were documented.

3.1.1 Functional Requirements

Prior to stating what the tutorial should do it was necessary to understand what it potentially could do. The research for this took two paths. Firstly the computer mediated communication section of the literature review highlighted the technologies involve in an online course. It was then necessary to take a step back and review some of the tutorials were currently available online these included:

- www.w3schools.com, a web standards such as html tutorial, which had a set of linear trails scaffold by source to screen examples,
- www.bbc.co.uk/gardening/htbg, a gardening tutorial, which again was very linear but supplemented the text with a range of multimedia presentations and evaluation exercises that on completion lead to accreditation from the Royal Horticultural Society,
- http://www.jimthatcher.com/webcourse1.htm, a web accessibility tutorial, which though lacks functionality related to e-learning, had good navigation and search functions.

It was then necessary to examine the way in which the tutorial was to be delivered so that the functionality could complement it. This entailed a short discussion with the client (project tutor) as to ascertain where the tutorial would fall in the online paradigm grid (fig 1.4). The outcome of this was that the client wanted to occupy as little staff time as possible and present a clear set of learning objectives but allow room for further reading by the user. This translates to a strongly learner determined, task specific approach to delivery. With this decided further correspondence with the client matched the approach to the final functional requirements

3.1.2 Non-Functional Requirements

The approach to highlighting the preferred properties of the site was slightly different. Many of the legislative requirements such as SENDA were highlighted in the literature review however to simply adopt these would be quite a minimal approach and so in order to gage user preferences as to the salient attributes of good tutorials two informal focus groups were held. The method by which these were conducted was a loosely scripted interview where they individually talked about an experience where they had used a tutorial online and how successful it had been. The first interview used a set of students of mixed course and year group. This was not successful as many of there courses did not use any for of e-

learning past placing materials on black board. In response to this a second focus group was formed using students that had, more specifically, learned to author html online. A computer with internet connection was used so that the interviewees could actually visit the online tutorials and highlight points. The points of discussion included:

- Which tutorials were good and bad
- What made them good
- What made them bad
- Why they would go back to a site to study another tutorial
- Whether it was hard to get started
- Was the tutorial easy to find

From this a transcript of the interview was produced and a set of recommendations produced. The interview followed the procedures laid out in (Dix 1997).

The three key non-functional requirements for this project are compatibility, usability and accessibility. These aspects are very interdependent. Accessibility is as much to do with usability as it is to with meeting the web standards within the compatibility requirements. Accessibility is a legal requirement and so is of paramount importance. In order to gather these requirements it was necessary to use the guidance of experts. Web tutorials and colleagues on the course were consulted in order to gather the requirements.

3.1.3 Documentation method

This information was then assimilated into a coherent document. This was not a trivial exercise as many of the requirements conflicted and it can be hard to distinguish between what is required and design solutions. A method was proposed by a college by which all the possible requirements were written down in the head of index cards. These cards were then numbered. Sub functions were placed in the body of the card and labelled and then the way in which these sub functions would be tested was inscribed on the back. Then the feasibility of each card can be assessed and conflicting cards were isolated and rectified. The card were then sorted into groups and compiled into the requirements document (Appendix E)

3.1.4 Summary

Now that the way in which the requirements were gathered has been stated, there will follow a discussion regarding the design decisions about conflicts and feasibility trade offs in the key functions and properties of the site.

3.2 Key Requirements

This section looks at the core functionality of system and how it is justified by the subject matter contained within the literature review and opinions gained from the second focus group. It also details and justifies amendments to the document.

The abstract requirements are seminal and expand on the software aim by incorporating the theory behind Kolb's learning cycle (Kolb 1984). If the system were to simply disseminate core content then the users would not be able to reflect on how well they have assimilated the knowledge. The client specified that there should be minimal staff involvement and this accounts for the evaluation being directed by the user. The final abstract requirement states that the system should facilitate interaction between users of the course and experts in ASP. This is justified in the literature review in the section on interaction. This type of interaction allows the user to query points if they are unclear. However this is only one form of traditional interaction. The design of the core content of the courses must include a rhetorical form of interaction into the textual monologue. This will perform the secondary role of interaction which is to keep the students attention.

The information about type and roles of users was ascertained through correspondence with the client. It was thought that the tutors, who may well be lecturers or researchers, could work on a voluntary basis and would do so in return for using the course to disseminate tutorials on their research in ASP. The characteristics of students identified, such as the management of their own progression through the course, directly influences other requirements, such as allowing a user to view all the tutorials stored on the system and simply highlighting which ones they have not completed.

One of the key functional requirements is the login and registration into the system. IT was decided this was necessary so you could moderate who was using the question boards, the system could store the user's progression through the tutorials and it gives the administrator the potential to contact the users to get feedback on the course at a later date.

Gaining the contact email address of users is the least important of the three reasons for registering users with the system. The initial requirements stated that the site would generate a password and this would be emailed to the users email address. However four out of the five users who performed the initial paper prototype evaluation stated, when asked, that they preferred direct access into a site after registration. If this was implemented however it would have probably utilised a PHP hashing function such as:

```
$password = mk5(mk5($username).sha1($email));
```
(8)

Instead of this the requirement now stands that the system must screen for incorrect email addresses. The design of this algorithm will be discussed later in the document.

Initially the system required a set of multimedia tutorials. This was due to account for a range of learning styles between students. Upon further inspection it was noticed that this conflicted with the accessibility and compatibility requirements. Various design options for displaying multimedia elements were explored and this will be discussed at some length in the design section. However once the system architecture and language had been decided it conflicted with requirement 6d, the upload of tutorials. These multimedia elements would have to be uploaded onto the site and the server had the PHP file upload configuration option on the server

was set to false. The combination of these three conflicting requirements meant that it was amended.

The data requirements were assembled on a basis that the datastore on the site should be a minimal set constrained by what is needed in order for the site to function. Hence extra information about the user or dates tutorials were produced were left out from the document.

Due to the nature of the software design process a change log for requirements was kept as an appendix to the document. This ensured that amendments were transparent and duly considered. This said there was also a conscious effort made to ensure that the requirements were concise and unambiguous but not so comprehensive that it limited the design options. This caused the document to remain quite stable with only a few minor adjustments being made over the course of the project.

The usability requirements are quantities however metrics will be placed next to them after the design is specified. These requirements are based upon and justified by the usability goals laid out by Dix (1997).

SENDA gives no specific measurable advice on accessibility of websites further than institutions are expected to make reasonable adjustments. There is no way that the design of the system will account for every disability therefore its 'reasonable' to bring the system into line with the current web standards. When the centre for accessible environments was consulted they pointed out that the Web Accessibility Initiative guidelines covers the working of a website but not the written content. Content can often be inaccessible just by the way it written. Requirements 11 and 12 were included because of this.

The distance nature of the course determines that it must be provided over the internet as a website. This will ensure the maximum number of people can use the site. The software and hardware that the website must be compliant with was based upon the statistics produced by the word wide web commission (W3C) as to demographics of browsers, hardware and operating systems. It was found later in the project that the reason that Safari, the browser provided by Macintosh was not included in these statistics was as it masks itself as Internet Explorer 6 (IE6) to the browser detection software. Requirement 6d was then added.

The minimum screen size was based on the students accessing the course from a desktop or laptop computer. This is because these computers were deemed to be learning environments whereas mobile technologies are not.

The learning objectives of the course were set as to give an introduction to the topic. It was thought, once the site frame work had been created, the tutors could submit tutorials that give a more indepth perspective. It was identified in the lit review that when teaching mathematics it is important to address misperceptions about concepts first this is why any of the objectives are prerequisite to the subject matter. Once this had been decided the learning objectives were based upon texts on the subject (Baral 2002), open knowledge on the internet and the proposals for an online tutorial on ASP (Costantini 2004). These were then taken to and agreed by the client.

3.3 Known Issues

This section though written with hindsight extracts the issues that appeared most poignant at the point the requirements were documented. The section will then go on to identify the issues that transpired to have the most relevance to the project.

The non-functional requirements of accessibility and conforming to web standards were seen to be salient to the success of the project. In reality conforming to web standards is not the be all and end all of producing a web site that's compatible with most browsers. Often the standards need to be broken slightly to get the best compromise in display between browsers. The reason why these were held with such a high priority was that the accessibility was a legal requirement and as this project is a first attempt at web design the conformation to web standards was an attempt to get it right first time around.

Another issue that was considered at length was the trade off between making an aesthetically pleasing site and maintaining download times. With the main title banner that was finally produced taking an estimated 6.2s to download on a $28.8kbs^{-1}$ modem it's easy to see why the design finally leant toward a minimal approach.

Another issue one the trade off agenda was whether to fit existing skills in languages in java or to learn new languages with the additional expense of time.

The final issue with the requirements was whether it was possible to get a host for the website. It transpired that this was probably the most trivial part of the project as the University are more than willing to create databases for student projects.

With reflection the biggest issue with the set of requirements was more do with the things that were not specified such as whether the site should be able to expand the number of tutorials or the precise web standards that would be used.

3.4 Summary

This section has detailed how the requirements were gathered, the amendments that were made to them during the course of the project and the trade offs that would have to be made in the course of the project. In the next section a high level design will be documented from including the design of the user interface and supporting data structure.

4. Design

4.1 Software Development Process

In order to create a usable web application it is generally agreed that user involvement and iterative design need to be incorporated into the design process. The waterfall model for software development has therefore been adapted to meet these principles and reflect the time scale for development within the project.

1. **Goal Setting:** The initial stage will be converting the software aim and functional requirements into a website mission statement and set of performance goals respectively.
2. **Design Alternatives:** Three lay-ups will produced and evaluated in house against other websites.
3. **Graphics Design:** Three style sheets will be produced using different colour schemes and graphics.
4. **Focus Group:** The three style sheets will be evaluated by users in a focus group.
5. **Paper Prototype:** Once a design alternative is chosen a paper prototype will be created and evaluated through a focus group of students and a usability inspection. These methods were chose due to their simplicity and speed.
6. **Cognitive Walkthrough:** Final paper product will be presented to 3 users who will perform a cognitive walkthrough of the product.
7. **Redesign:** A formal GUI will be specified alongside dimensions of elements, error messages and content emphasis.
8. **Implementation:** A partial product will be produced using xhtml and a skeleton grey scale style sheet. This will be connected to a database and be published online. It will be iteratively tested through online validation, accessibility tests and black box testing.
9. **Task analysis:** The functional requirements will be tested using a task based analysis of the product.
10. **Implementation:** The lessons plans will be implemented then iteratively tested on a range of platforms and browsers.
11. **Final User Testing:** The initial focus group will return to be observed using their design. There performance will test whether the system has met the performance goal and their reaction will validate the site aim.

4.2 High Level System Architecture

Requirement 14 states that the software must be accessible online and conform to the w3c web standards. The design question remains however as to which web standards to adopt. As a base standard the site should conform to xhtml version 1.0 and use a cascading style sheet for lay-up. These standards are compatible

with the required browsers and adopting the xhtml standard will ensure the mark up is fairly future proof.

In order to fulfil the functional requirements and meet the data requirements [req. 8] it is necessary to store data about the users of the system. Prior to specifying how the data will be stored and accessed it is necessary to look at the functional requirements and assess who needs access to the data and what data they need.

	Login data [req. 8a]	*Course progression data [req. 8b]*	*Questions [req. 8c]*	*Responses [req. 8d]*
Users	Need Access to there own data and no-one else's	Need Access to there own data and no-one else's	Need to access all question posted as not to ask repeated questions	Need to access all responses
Tutors	Need Access to there own data and no-one else's	No current need for data due to self assessment	Need to access all unanswered questions	No role related need to access data
Admin	No current need to access data as login has a procedure to re-register students [req. 5.e].	No current need for data due to nature of the role of the administrator	No current need set in requirements	No current need set in requirements

The information used within this table can now be used to assess the best way of storing the data required. There are three commonly used methods for storing data used in web applications.

Data relating to the individual can be stored on the users' memory in the form of a *cookie*. Seen as currently the requirements state that the only user needs access to information on their progress through the course cookies could be a valid way of storing this data. Cookies can also be used to identify users instead of a login option. Unfortunately this is not suitable here as the user would not be recognised from different locations and other functions are dependant on the login information. Cookies are not appropriate, however, for information that needs to be shared such as questions about the course and the tutors' responses for them. On reflection, to store any of the data using cookies represents a lost opportunity to centralise the store and add the potential of extra functionality. This combined with the privacy issues linked to the user acceptance of cookies leads us to look toward other methods of storage.

From this point on it is assumed that the data used in the application should be stored server side to ensure that it is sharable and so not to restrict further development of the system.

Data used in web applications can also be stored in a separate file in *xml* of extensible mark-up language. The web browser would then use client side scripting to modify the data based on the user inputs. The most commonly used client side scripting tool is known as java script but 11% of web users choose not to enable the java script as to avoid pop up advertisements. Xml was designed to

represent and facilitate the exchange of data across different platforms and though the data on the system needs to be sharable between other users of the system it would have little use to other web applications. On top of this, the xml standard was not designed to represent relational data as it nests elements within a single root element in a hierarchical manor. Metadata, such as primary keys, could be placed as attributes of tags but there is not a wealth of freely available software tools to extract such data. All of this means that the adoption of the xml standard is not ideally suited to the nature of the system under consideration.

The system being developed will be the property of the University of Bath and hence the final remaining option is to use a server side scripting language and DBMS that is supported by the University's servers. The university uses a MySQL database server on a UNIX platform. MySQL is based upon structured query language which aptly allows us to define relational databases. This data then needs to be scripted in to html using a server side scripting language. The university servers use PHP or PHP Hypertext Pre-processing which is free to used and when combined with MySQL it provides a web database application development tool. This makes it the idea standard to meet the data requirements of the project.

The requirement document also specifies that "Students should be able to view a range of multimedia presentations" [req. 6a]. The final design decision, with regard to compatibility and standards is the way in which multimedia elements of the site are displayed. A multimedia presentation can consist of text, images, sound, animation and video clips. There are various ways in which this can be displayed online.

Firstly the presentation could be displayed as a video clip combining the other media elements using an editing suit. There are multiple video formats with equally varying properties. The video would have to be downloadable across a low bandwidth [req.18] and hence it should either be streamed or compressed. It should also be available across platforms and if it requires a plug in this too should be widely available. This really limits the choice of format to mpeg, QuickTime and real video. Editing suits for mpeg are free to come across but the end product takes time to download even on fast connections. The free QuickTime editors only allow you to sequentially place together audio video interleave files and other QuickTime files this doesn't allow the inclusion of sound and text information. Real video files seem to solve the band width problem as they ca be streamed however the resolution is so low that text information is hard to read. This leads to the conclusion that to deliver the multimedia content as an inline video format is not feasible. They are generally slow to download, hard to represent text and don't cater for interaction.

This said the design is left with two final options; flash and java. Java is really well established and doesn't require a plug in. It is scalable to any processor driven platform and interacts well with other web elements. The skills are already present to implement the design and there is an array of online resources to support its implementation.

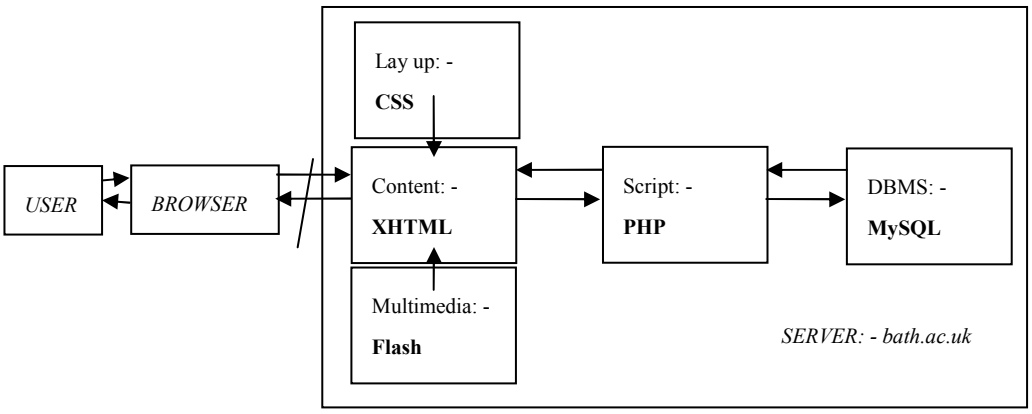
Flash is also now widely available but less well established. It requires a plug in but this comes as standard on the required browsers [req. 14]. It supports a wide

range of graphics (GIF), sounds (mp3) and functionality (transparency). It is self contained and so has faster download times and better playback rates over a variety of platforms.

The question remains, which is best for the Logic Programming Tutor? The presentations do not need to be integrated with the rest of the system. A higher download and cross platform interoperability is preferred. Though flash depends on a plug it is now broadly used (96.4% Neilson). Though Java is requires less training to implement and is operable on a range of platforms it seems flash will be adopted due to download speed and range of media it supports.

As the university server does not support file uploads from PHP however a system will need to be put in place by which the flash scripts required by tutorial are emailed to an administrator who the virus checks them and places them in the file space associated with the tutorial.

It is now possible to depict a standards based abstract architecture for the system:



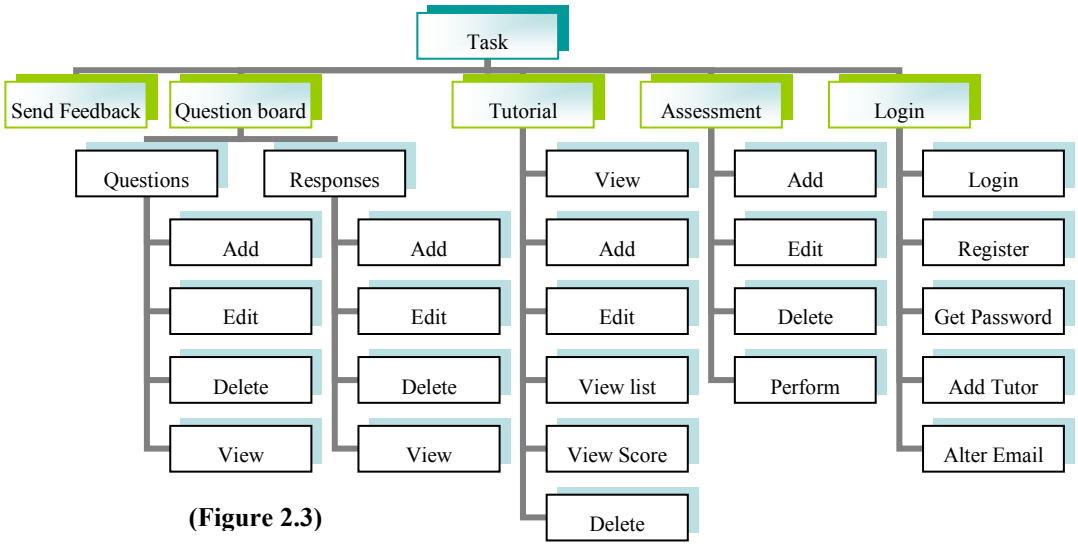
(Figure 2.1)

4.3 User Interface Design

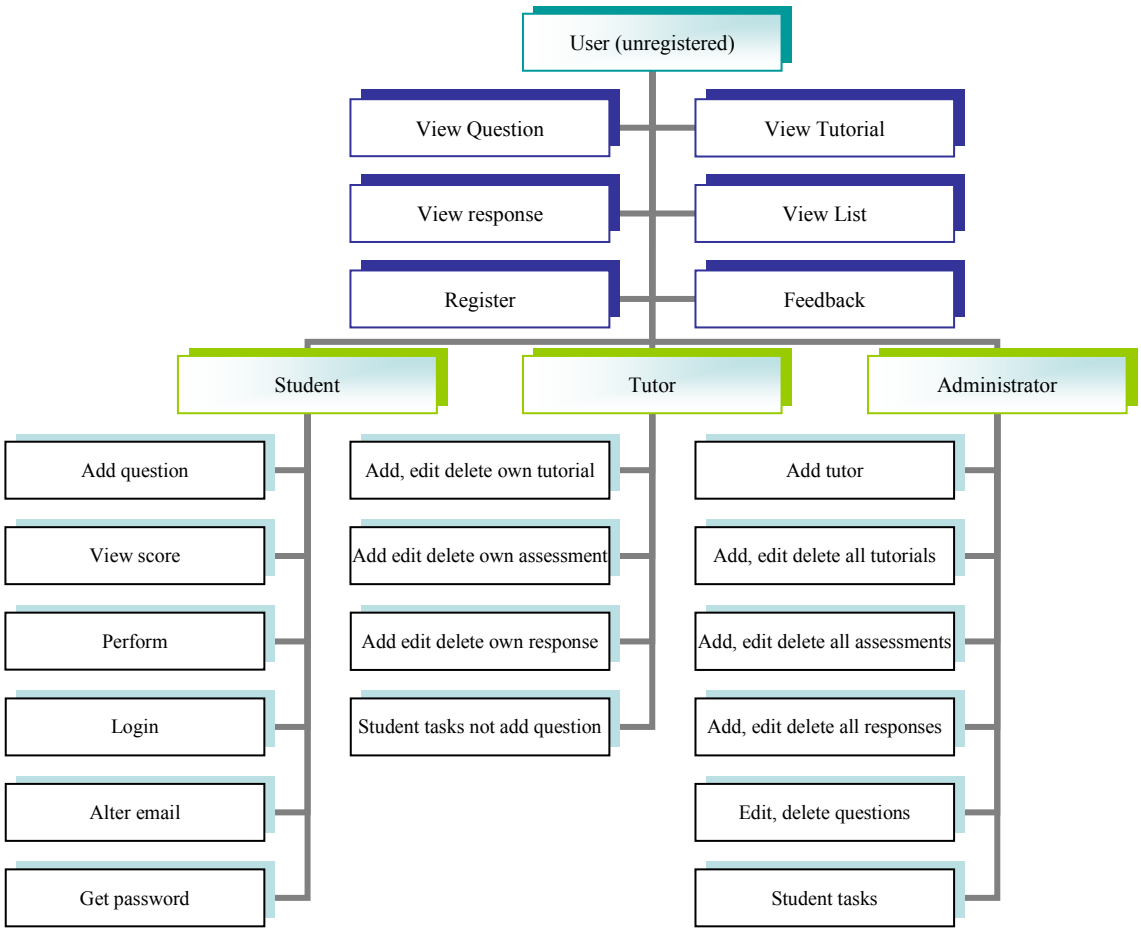
This section will break down the functionally of the site using various diagrams, then review the process of paper prototyping and finally discuss how the heuristic guidelines were collated.

The functional requirements can be broken down into a user task level. This is shown in figure 2.2. This then depicts the full range of tasks that the system is able to perform. Users registered with the site also need to be deleted; however this be performed through the MySQL database management system (DBMS) for security reasons. Instructions of how to do this can be found in appendix C. No the scope of functionality has been defined it is necessary to allocate these functions to different types of user of the system. This is represented in figure 2.3.

These diagrams represent amended versions of the originals which did not contain the option to expand the set of tutorials and assessments. Alongside the original models rough UML sequence diagram notations was used to identify the flow of information, sequence of functions and parameters passes in and out of them.



(Figure 2.3)



The usability design guidelines that the system will be tested against need to be explicitly stated prior to designing the user interface. A set of guidelines have been compiled from a range of sources including:

- http://www.useit.com/ Jacob Neilsen “usability guru” – quite vague

- http://usability.gov/guidelines/index.html US government's site – practical and thorough
- http://www.w3.org/TR/WAI-WEBCONTENT/ - World Wide Web consortium's recommendations on accessibility
- http://www.jimthatcher.com/webcourse1.htm Tutorial on reasonable adjustments to a site to conform to DDA and US article 508

The final list of guidelines can be found in appendix F. The list does not include any guidelines that were deemed to general as this should be dealt with in the usability goals statement. The list is far from comprehensive but it gives the design a sound footing. It's composed of six sections containing guidelines and where necessary justifications. The final design may not conform to all these guidelines but deviations from them will be justified.

Once these heuristics have been stated a paper prototype needs to be developed. A set of design ideas were produced, then validated against the body of guidelines. This set was then reduced to three alternatives, which can be found in appendix G with the paper prototype. Of these three, two have the navigation on the left hand side of the screen. This is not recommended in the guidelines. However it was deemed that the style of user interaction meant it was not necessary to place this bar in hot space.

It transpired that it was not possible to chose between two of the lay ups alone so the graphics design task which had originally planned to be done before the first implementation phase was brought forward and 6 cascading style sheets were produced. One for each of the two designs, each with three colour schemes. These were taken to a focus group of users who chose the lay up three. More is said about this task in the section on style in the detailed design and implementation section.

From this lay up a paper prototype was produced, using the aforementioned rough UML data sequence diagrams and submitted for a cognitive walkthrough. As the site is designed for undergraduate it was quite simple to find test subjects. The decision to perform a walkthrough at this stage was taken as so no to spend time authoring a website that fundamentally was not usable. Efforts were made to rapidly prototype using a package called python card however it transpired that it was quicker to produce the prototypes on paper. Many usability experts prefer to paper prototype as the user often feels more comfortable to make alterations to the design (Preece, Rogers et al. 2002).

The process of the walkthrough went as follows. The paper prototypes were placed on a cork board with string connecting links within the site. The users were asked to perform a series of tasks which directly related to the functionality of the site. The functionality scenarios listed in appendix A are roughly based upon these. The points at which the user could not understand what to do next were noted. The users were asked to list what they found good and bad about the site and how it matched up to the usability requirements in appendix E.

From the scripts of these interviews, a set of labelled scale screen designs were produced. These were the specifications by which the web pages were authored.

Unfortunately these cannot be included in the final document as they were damaged in a household flood; however in their place now stand the original paper prototypes. Various recommendations came from these walkthrough including:

- Title bar to span the width of screen,
- Registration and login to be in the side bar,
- The message boxes are java script and some users have this turned off,
- Lessons should come before questions in the function bar
- Contact Details can be on the front screen
- There should be one tutorial per screen

4.4 Database Modelling

After the design of the interface it was necessary to model the back end of the application, the database. The system is required to store the following information [req. 8]:

a. User-names
b. Passwords
c. e-mail addresses
d. user types (administrator / student / tutor)
e. Tutorials users have viewed
f. evaluations completed
g. scores obtained
h. Questions posted to the question board
i. User name of the student who posts a question
j. The date each question is posted.
k. The tutors response to the question
l. Name of tutor who posted the response
m. the question it was in response to
n. The date the response was posted.

From this the following entities can be identified:

Users of the system (Username, Password, User type, E-mail Address)

Tutorials Completed (Username, TutorialNumber, Tutorial Name, Score, Completed)

Questions and responses (Student Username, Question, Date Asked, Tutor User Name, Response, Response Date)

This is in first normal form as all of the columns are singled valued. It is not in second normal form as Tutorial name is dependant on tutorial number and that is a component of the key. This can be amended:

Users of the system (Username, Password, User type, E-mail Address)

Tutorials (Tutorial Number, Tutorial Name)

Tutorials Completed (Username, Tutorial Number, Score, Completed)

Questions and responses	(Student Username, Question, Date Asked, Tutor User Name, Response, Response Date)

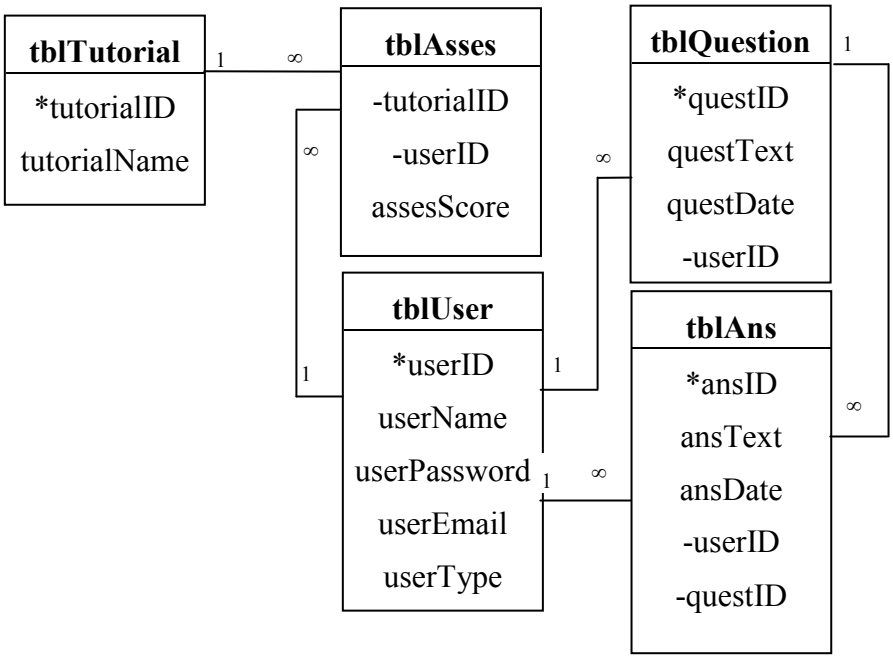
To move the model into third normal form the non-key attributes must not be facts other non-key attributes. The response date is a fact about the response and so is the tutor who responded. In third normal form the model looks like this:

Users of the system	(Username, Password, User type, E-mail Address)
Tutorials	(Tutorial Number, Tutorial Name)
Tutorials Completed	(Username, Tutorial Number, Score, Completed)
Questions	(QuestionID, Student Username, Question, Date Asked)
Responses	(ResponseID, *QuestionID,* Tutor User Name, Response, Response Date)

In addition to moving into third normal form non-meaningful identifiers have been added, as there is a rare chance that two people may ask exactly the same question. Question ID has been placed in the responses entity as a response is dependant on a question being asked. This also allows for various tutors to respond to a question if they so wish. Due to the lack of any multi-valued dependencies the model is now in fourth normal form. The only point that remains is how to reduce the use of memory. There is no need for the "completed" attribute as it is a consequence of the existence of the instance. Username could be supplemented by a non-meaningful identifier to save some space if names are long. The final model can be seen below.

4.4.1 IDEF1X Model

Primary keys are highlighted with an '*' and foreign keys with a '-' symbol (though foreign keys don't need to be defined in MySQL as the JOIN statement suffices).



4.4.2 Data Type Definition

Within the implemented database the identifying attributes will be small unsigned integers as the site will not have more than 65,000 users. Identifiers where there are a limited number of instances will have a tiny integer data type. The primary keys will be automatically incremented to save scripting. The other non-identifying attributes will have the following attributes:

Attribute	*Data type*	*Justification*
tutorialName	varChar(100)	Maximum 10 x 10 letter words, variable column width
assesScore	tinyInt	Unsigned max 255, score will be calculated as percentages.
userName	varChar(16)	6-16 characters standard on the internet.
userPassword	varChar(16)	As above
userEmail	varChar(100)	Well above current longest email address
userType	enum('s', 't', 'a',)	s for student, t for tutor and a for administrator,
questText	text	Next biggest after 255 characters. 65535 is really to big but 255 characters it too limiting.
questDate	date	Only date format (YYYY-MM-DD). No need to store time of post.
ansText	text	Sannme as for questText.
ansDate	date	Same as for questDate

Using this data and the IDEF1X model the MySQL Database supporting the web application can be implemented.

4.5 Summary

This section has specified the high level system architecture, the specific standards that will be used in the system, the user interface, the data model and the process by which the final system will be implemented and tested. The next section looks in detail at the problems and nuances of the final design, design functionality that was attempted but failed and in particular the scripts that glue the system together.

5. Detailed Design and Implementation

5.1 Approach to authoring the scripts

Initially the screens were authored using xhtml, laying out divisions, placing comments where PHP scripts would potentially be and stating there functions involved in the scripts. This reflected the order in which the standards used were learnt. This also gave a scope as to the magnitude of what needed to be coded. The PHP scripts were then inserted one by one in an order based on the functionality of the site laid down in the requirements document. Each function was broken down into a list of subtasks. On completion of each sub task the code was tested and once all the subtasks within a function were complete the section was tested by a user. This will be discussed in greater detail in the testing section.

The code is required to meet the standards presented by the W3C. Further than this however it was necessary to maintain some sort of order to the pages. A synopsis of the page's purpose was commented at the top, followed by functions performed on information passed to the page and then code that would be displayed irrespective of the state. When the success of an interaction with either the database or the user was tested, failure would be placed first. This meant that if the code nested multiple tests it was easy to associate an error message with a test and hence the purpose of the test is made much more apparent.

It was deemed important to clearly comment the code so in addition to the synopsis of the pages function addition comments were place before any function calls, conditional statements, database queries and xhtml forms.

5.2 Scripts

The code that was finally implemented can be found in appendix d. It is commented and so should be faily intuitive as to how it works. In layman's terms the code defines what is shown on one page in the site at any one time. The system alters the page being displayed in the browser on one of two events. Either when a hyperlink is clicked (9) or when a form is submitted (10)

```
    <a hfref=pagename.php?state=somenumber>textonpage</a>          (9)
```

```
<form method="" action="pagename.php">                            (10)
       <input some form element  />
       <input some form element  />
       <input some form element  />
        <input type="submit" /> //the button
</form>
```

When one of these events occurs, the system loads information into either the GET or POST array. The next page first examines whether any information regarding the state of the system is in these arrays and then controls what is displayed based upon this using 'if()' statements.

Information is placed in the GET array if it is appended to the page name of a hyperlink after the question mark or if the method of a <form> element is “get.” The information is placed in the POST array only if the form method is “post.” All information placed in the GET array is stored in the url of the page and so is therefore visible in the address bar of the browser. The system places information in the get array if it relates to the navigation of the site so users can book mark pages. Only information that is personal to a particular user is placed in the POST array.

Forms occasionally refresh the page that is being displayed in light of new information submitted by the user. When this occurs, the form action attribute is <? PHP_SELF?>.

The final design decisions relating to individual functions will now be discussed. Further explanation of the code should be self evident from the commenting.

5.2.1 Login

The second focus group during the requirements gathering phase highlighted the fact that they often lost there place in a online tutorial when they performed a support function such as registering on the site. For this reason the login function contains multiple forms that in other sites would be passed to separate pages, in a single script. This allows other functions to maintain state while a user is logging in or registering.

The whole login and registration script is stored within the file phpfunctions.php. This is because it is called at the start of every page. The design of this function was quite taxing. Often several forms would lead to the same point in the process and so it was difficult to translate the paper prototype into a script without repeating the code for the same form multiple times. To minimise this, a process was developed where by forms were mapped to a minimal set of functions. These functions were then mapped to a set of outputs they could produce. Then these outputs were mapped back to the forms they displayed. A finite state diagram was produced where by each form occurred only once. Then all the conditions for the form being displayed were listed and a control structure could be coded.

It was found that PHP tests all of the clauses in a conditional statement so for example;

```
if(($x ==1)&(y==2)){
```
(11)

Would test both the x and y conditions even if the x condition was false. This meant that the functions could not be placed in a single conditional statement with the state test that induced them as they would get called twice. This meant that the design outcome of the finite state diagram was not quite implemented as some nesting was used in the control structure.

Error messages supporting the login and registration process were placed inside the functions in order to reduce the number of possible forms. This in turn meant that the messages could reflect that actual problem not just the process that was failing.

5.2.2 Sessions and Security

The control of access to functionality of the site is controlled by the use of session variables. These are stored in another PHP array known as the SESSION array. The system works as follows. At the start of each page a session is started. Variables can be posted to this session. If variables were posted to a session in the previous page then the session is continued from the previous page. Otherwise a new session is started. Variables have to be registered with a session before values are added to them. The session used in the site security stores the user name type and id. These variables are registered on login. They are then used to control the links and forms that are visible to the user and hence which functions a user has access to.

Sessions are particularly good for controlling user access as, unlike cookie, the variables are stored server side. A unique id for each user is stored in the browser and this is used the access the users session variable. This method is recommended in the PHP documentation as the most secure method for controlling data access.

5.2.3 Questions

The concept of maintaining the position the user is in within a tutorial was extended to the question board. Though the initial interface mock up stated that it should be contained on a separate page users found this mean that they would loose their place and were not inclined to use it as the tutorial progressed. To amend this it needed to be opened in a separate window. The DTD chosen for the site was xhtml strict as the site was using divisions not frames. This meant that the target attribute of links was depreciated. The only other option was to use java script. If the user's browser was not java script enabled then it would display question_boardns.php (question board no script) other wise it would call a java script window on the onclick attribute of the link in the new top window.

While implementing the question board it was noticed that if the user input a string with comments of MySQL key words in it then it would not be stored but the query would return true. This error took a while to uncover and to account for this the add and strip slashes PHP MySQL functions are called before anything textual is added to or retrieved from the database.

User testing highlighted that the initial user interface for the question board was not usable. From this a back to top link was added and the question and tutorial sections were segmented using borders in the style sheet.

5.2.4 Tutorials

Once the decision was made to introduce an "add new tutorial" function that would be available to site tutors the focus was taken away from producing a range of tutorials to gaining a working upload script. A single tutorial was produced which is currently stored on the database. This tutorial attempts to set the tone of future tutorials. It follows a format gained from reviewing literature on teaching methods.

The tutorial initially states what the students will learn then breaks down these objectives into various headings and addressed each one at a time. Supplementary links were given as links to outside websites. The last section reviewed what was taught then pointed the user in the direction of a self evaluation. Mid-way through the tutorial the student is set a rhetorical exercise of "can you find other applications of ASP on the internet?" This is a form of interaction that regains the student's attention and points them in the direction of work outside the lesson.

When it was realised that range of tutorials need to grow from those offered in the requirements some significant redesign work needed to be done midway through implementation. The tutorials entity within the database had two extra fields added. These containing the source code for the tutorials and assessments. Forms and extra scripts were added to allow for tutorials to be added, deleted and edited and new pages were added to the file structure. If the PHP upload file option had been set to true then the source code could have been transmitted as a file.

This requirement really should have been picked up early on in the development as it gave rise to other design options that given time could have increased the utility of the site. The resulting design included a text area upload form for the new tutorial html and a read me file with suggested layout guidelines to ensure tutorials submitted were up to standard.

In the design of these templates it was realised that the tutors could not post equations on their tutorials. This was because the only method for displaying equations in html supported by all the browsers listed in the requirements document is to create the equation using another package then converted it to a GIF and place a phonetic description in the alt tag for accessibility. These images would then have to be emailed to the site administrator and place by hand on the file space. This is the method supported by the submitted code.

During the course of the design implementation a substantial amount of work went into automating the above process. It was thought that a tutor could type in the equations in tex format in the code and a PHP script could automatically convert it into an image format on the site (see 12).

```
<div class=”equation”>
/fract{/alpha}{/beta}
</div>
```
(12)

As a point of interest the design for the script will now be discussed alongside reasons for it not being implemented.

A script was first designed to remove the division of the equation and replace it with a image tag. To do this the preg_replace_callback php function is used.

```
preg_replace_callback(
     '|<divclass="eq">(.*?)</div>|is','convertEquations', $html
);
```

The first parameter signifies that the text between the divisions is the string that we require to use. The second parameter is the name of the function the text will be sent to and the third parameter is the initial code that needs to be converted.

Once this is done the 'convert equations' function would perform the following tasks

Take the 0^{th} item in the array that is sent to convert equation as this is the complete match. Other items in the array are sub pattern matches. With this the following functions need to be performed:

1) Add the latex document class begin end and equation tags using the string concatenation function '.'.
2) Save it in the same folder as the webpage as a file using the fopen() function, which will create or overwrite an exiting file and fwrite() functions
3) Use the execute() function to run the final commands on the file created:
    a. latex2html
    b. copy the image created from the subdirectory it has been placed in to the current directory
    c. delete the subdirectory
    d. run the mk5() function in PHP passing the latex that was extracted from between the tags to it to create a unique code for the image.
    e. Rename the image to that code.
    f. Run a giftrans program on it to make it the correct size
4) Return an image tag with the unique code as its href attribute and the latex as its alt tag.

The source code would the display an image tag between divisions (see 13).

```
<div class="equation">
 <img href="0012355423.gif" alt="alpha /backslash /beta"  />
</div>
```
(13)

5.2.5 Assessment

A similar set of upload forms, scripts and read me files were created for the creation of an assessment for a tutorial as were for a tutorial. The difference between this and a tutorial is that the tutor is asked to find the points within the code where multiple choice questions and answers are added and alter the questions to suit their tutorial.

The script for calculating the score and updating the database is also contained within script. On reflection this may pose a small security risk by allowing the users to see part of the data structure behind the site. This said the only users who will view this file are tutors and they are personally validated by the site administrator.

The assessment source code is stored within the tutorial entity. This is not very adaptable as it only allows for one assessment per tutorial a much better structure would have an extra assessment entity with tutorial id as a foreign key.

Both the html entry methods for the tutors to add new tutorials and assessment are quite complicated. The user requirements state that the tutors should have some knowledge of html but there is probably a simpler way of transmitting this information. If the site was to be developed further then the tutorial and assessment forms could be transmitted in xml and use a schema to ensure they are in the correct format. This would constrict the site to text and equations but in turn this would mean that the site administrator could spend more time checking the academic standard of the submissions as opposed to uploading files onto the site.

5.2.6 Feedback

The feedback function was not listed as a requirement however it was noted as best practice on both Neilson's and the W3C's website and so it was included. The script was a trivial function called mail() in PHP. This is better than the "MAILTO "attribute of the anchor tag as it bypasses any email software and sends the message directly. There were some problems placing a reply to address and so the reply to address was placed in the body of the text.

On the subject of email feedback the address of the staff member responsible for the course is placed at the bottom of each page. During a user testing session on of the users highlighted the fact that this email address would be subject to malicious spamming software that trawls the internet searching for addresses. Through fear of professional misconduct the text was immediately replaced with a gif image of the address with an alt tag spelt phonetically. No sooner was the amendment made as it was realised that that particular email address is freely available on the universities person finder. Needless to say the other pages were not updated however the original gif remains as a demonstration of a satisfactory method for avoiding spamming.

5.2.7 Style

The website was authored in xhtml. Therefore content was clearly separated form style. It was deemed necessary to define the style document for the site at an early stage of the development. It was understood that in reality the style and content of a document are closely intertwined. As extra types of elements are added to the site more CSS classes need to be added. A rough style sheet needed to be produced early on in the project. Hence, as soon as the paper prototypes were agreed on, training in CSS started and a style sheet was produced.

It was required that the site operates on a low bandwidth therefore the more graphics a site has the slower the download time. The paper prototype was inspected and two style related graphics were identified. These being the information in the header and the background of the navigation bar.

The style needed to be based upon the site content. Ideas were quickly brainstormed and two key attributes were identified. These were that the site should convey how contemporary the field of study was and yet still remain down

to earth so not to deter users. With this among other things in mind a list of sites that had a style similar to this was compiled including;

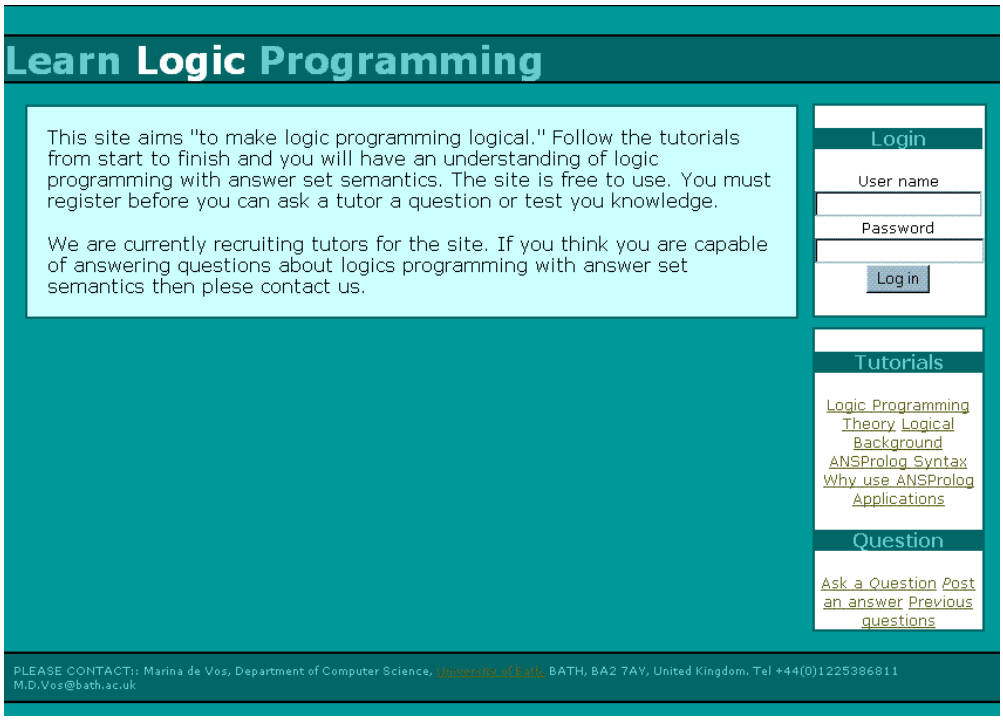
- http://www.chameleon-web.co.uk/
- http://www.bbc.co.uk/
- http://www.moma.org/
- http://textbased.com/
- http://www.macromedia.com/

All these sites used sans serif fonts such as Arial (Helvetica on mac) or Verdana. They frequently concatenate the title words and distinguish between them using different type faces. With these features considered a two greyscale designs were produced.

The colour scheme used the intensity of the grey scale and with the hue of the colours to ensure that users with colour-deficiency could distinguish between them. A colour picker was used as the Author, himself, is colour-deficient. The three schemes chosen reflected opposite, adjacent and contrasting colour combinations. The schemes had no more than seven colours (Schneiderman 1998) and the non colour wheel colours should comprise of buffer colours such as whites silvers creams and greys. The colour picker used can be found at

<Colour Theory http://www.colormatters.com/colortheory.html> (14)

These designs and colour schemes were shown to a range of students and feedback was noted. In the end a greyscale colour scheme was chosen with a left hand navigation bar. The rejected design can be see in figure 4.1.



(Fig 4.1)

The CSS document is standard CSS lay up. There were attempts to make the tutorial menu a drop down vertical one using nested lists. This made use of CSS pseudo-classes to the tune that the ‘onhover’ attribute of the parent elements within the list cause the child elements visible attribute to set to true. This was implemented but was found to have positioning errors when view on various browsers.

After the design was completed the University of Bath a statement including guidelines on web visual identity. It was deemed too far into the implementation to alter the design (10). However as the site revolves around a separate style sheet it would be fairly simple to give the site a University of Bath corporate image.

<http://www.bath.ac.uk/pr/visid/visidreg_web.htm> (15)

5.3 Accessibility Issues

The accessibility features that the site offers are often a matter of not making the site inaccessible as opposed to making it accessible. The choice of xhtml as the mark up language ensured that style was separated from content, no frames were used and the page was set out in meaningful divisions as opposed to table format. This allowed the screen reader to easy read the content of each screen. When there were long pages such as the question board the page was navigated by internal links. This is far more relevant for someone accessing the site in aural format. Internal links are used to negate the fact that the screen is laid up using a left han navigation bar. There is an invisible link in the title division that lets a users listening with a screen reader skip to the main content of the page as opposed to listening to every link on the page being read out.

Colour plays an important part in the accessibility of the site. The site is predominantly in greyscale. This ensures that the site can be viewed by student with colour vision disabilities. The only other colour used is red this denotes error messages but it's the content of the messages not the colour that's important. The foreground and background colours also contrast to ensure partially sighted users have accessibility.

By following the usability guidelines the site that has been produced is accessible. The extent of this will be tested in the next section

5.4 Summary

This section has looked in detail into the design of what was finally implemented. The next section will test the design assumptions made and check the final software product for errors.

6. System Testing

6.1 Testing Strategy

The method of incremental testing as functions were complete that was mentioned in the script authoring section meant that the majority of faults in the system were detected during implementation. Involving a user in a walkthrough of each function after completion also meant that the site to some degree was already usability tested. This section really addresses the summative test of whether the system meets the requirements stated at the start of the project. The section describes the tests that were performed and the justification for them the results of each test can be found in appendix D.

Tests are grouped into sections that reflect the types of requirement within the requirements document. The documentation of each subsection will discuss the methodologies used, the scope of the tests and the resulting unresolved problems with the system.

6.2 Functionality Testing

The functional requirements have to a strong degree already been defect tested (Sommerville 2002) in the course on the implementation. This said there are still some tests that can be performed. Each of the functions needs to be tested to ensure that the requirement has been fulfilled. The requirements will be address consecutively and site will be inspected to ensure that under normal conditions they are operational. The functions will then be tested under stress; firstly by inputting extreme values of both size and magnitude of data into every form on the site and secondly by allowing a group of proficient coders in PHP to attempt to break the code.

This testing will be limited to a finite number of browsers and operating systems. If it were to be performed comprehensively then it would need to be performed on all the browsers and platforms listed in the requirements documents. Unfortunately the number of permutations this creates vastly out weighs the resources of this project.

After the testing was completed, it was realised that the late addition of the add slashes function had caused a fault in the system which made the assessment upload not display correctly. This coupled with several smaller previously undetected faults means that the site has not met two of its functionality sub requirements. This will be discussed further in the conclusion

6.3 Usability Testing

This will involve three tests. Firstly each page of the site will be tested against the usability guidelines stated prior to the interface design, which are found in appendix F. The site has already undergone a task based analysis using the tests described in appendix A. This process also involved three interviews that ensured that the usability goals stated in the requirements were achieved. To measure the efficiency of the interface the users were timed in performing the set tasks and then asked if they deemed this a reasonable length of time to spend performing this task.

Usability testing is a very qualitative process and it is highly unlikely that the interface will meet everybody's needs. However, the tests undertaken coupled with the constant user involvement have ensured that the site is quite usable.

The usability tests were successful the usability goals were met and the usability guidelines were to some extent followed but it was realised that this form of testing is not a good summative method. It would have been better to get users to list changes early on the in project and then perform user acceptance testing that the site was to the standards they stated.

6.4 Accessibility Testing

The Web Accessibility guidelines contain enough scope to make the site WAI compliant however some further tests are listed in appendix A. These are designed to be quick and easy to perform instead of working through the entire WAI guidelines.

The accessibility tests were all successful apart from the evaluation of the site using Bobby which was not available for download. The site was also inspected by a Student Union Officer who has lobbied on the SENDA legislation. They believed the site had made the reasonable adjustments required.

6.5 Web Standards Testing

The site will use an xhtml validation service to ensure the standards for each page have been met. This presents a problem however to the tune that the pages rely on the state input from the post array and as such will not be thoroughly tested. In response to this the site will be inspected on all the required browsers and platforms.

The web-standards testing tasks were an overall success. A few of the pages require the occasional paragraph element adding but on the whole the stand of mark up is high.

6.6 Summary

Unfortunately the site has not met all of the requirements set out at the start of the project. This said in other areas it has excelled. The data, usability, accessibility, and compatability requirements all return positive results and with a little more work on fault detection the site will gain full functionality.

7 Conclusions

7.2 Implementation Methods

The process that was stated at the start of the design section was a tried and tested software development cycle. Therefore it is wrong to assume that the reason the functional errors were not detected was the fault of the cycle alone. There must have been other aspects of the methodologies of development that caused the failure to meet requirements.

As with many software project the implementation over ran into the testing time. This meant that testing was being performed on an ad hoc basis after each sub function was produced as opposed to formally once an entire function was finished. There was no clear documentation method for testing. Sub-functions were listed in the order they were coded then errors that occurred were amended to the end of the list. This meant that coding occurred when errors were still present on the system. As the development drew to a close these lists, now predominantly filled with faults, came from a more diverse set of functions and so emergent errors were more likely to occur.

A part from having a more structured approach to coding and documenting tests, the error may have been averted by allocating test time to design and more time to testing.

This was the first time the author had developed a web based system and so large amounts of time were devoted ate the start of the project to training in web standards.

7.1 Development Methodologies

The non-functional requirements can contribute the success to a sound design developed early on in the design. The extra training time allowed the pages to be coded to standard right from day one. In some respects the fact that this was a first attempt at web authoring may have been beneficial to meeting the web standards requirements. This is due to the fact that there were no bad practices learnt from previous releases of the web standards such as the use of tables for lay up and using fonts in content.

The initial involvement of users in the design really shaped the outcome of the site and ensured the usability requirements were met. This said there was really no documentation of this process apart for comments on screen designs and transcripts of meeting. If the project were to be repeated then it would be advisable to use a set group of users through out the design and implementation process. This means that metric could be take regarding users completing tasks in order to build a base of empirical evidence to show that the usability of the sight improves as the design develops.

7.3 Research Processes

Two research methods were used during the course of the project. In order to train on web standards tutorials were taken at regular periods throughout the projects on a range of standards much wider that the set of those actually used. Paper notes

were produced for every tutorial and filed for later reference. This was supplemented by reference to the documentation for these standards and an example library was built to reflect aspects of the sites functionality that would need to be coded at a later date.

The second research method was not as successful. A range of academic literature needed to be digested in order to produce a set of tutorials. This was found to be very difficult to do. This is the real justification for focusing on allowing tutors to submit there own tutorials. The method used to deal with this problem involved reading and annotating vast numbers of papers, books and websites. The lesson learnt is that if this point was reached again it would have been wise to ask for assistance.

That said experts were consulted on a range of other issues from posting questions on php web forums to access consultants. If the project were to be repeated then a range of expert would be identified right from the start of the project so time is not wasted midway looking for assistance.

7.4 Unresolved Problems

The key problems encountered during the development of the site have been mentioned to some length already however as a point of summation they will now be recapped. The largest amount of work that was lost due to simply not being able to solve the problem was the conversion of latex to gif images in img tags. It was really disappointing to get so close to a solution and not be able to use the key program (latex2html) due to a lack of technical support and clear documentation. This said the university of Bath web team are still interested in the problem and so there is a strong chance a solution will be produced by the end of this academic year.

The final two problems encountered quite were out of the range of control of the project. Firstly, the file upload was disabled on the University of Bath's servers stopping new tutorials being posted to the site. A solution to this was to upload text through a form but this transpired to be insecure and prone to errors. A better solution is proposed in the section on further work.

Loss of information occurred when a pipe burst and destroyed a full set of paper prototypes at a late stage in the project. This was unfortunate as the prototypes were not reproduced. This is not the route cause of the problem however and points to a far deeper routed problem in the organisation of the project. The fact remains that there were no copies made of the prototypes as there was no formal testing process or documentation prior to the summative evaluation. This situation would be rectified in future projects by formally specifying such systems prior to the requirements definition.

7.5 Further Work

A few rectifiable errors still exist in the site and can be found in the testing section of the appendix. This would be an obvious step for further work. If the concept of the tutorial were to be expanded further then the aforementioned xml schema for submitting tutorials would be produced and tutorials could then be submitted via e-mail or over a form. Though the schema would restrict the what could be

displayed on a tutorial it would make it much easier for web novices to submit code.

7.5 Achievements

The key success of the project was producing a site that was firstly usable then compatible with a range of browsers and operating systems and finally conforming to the SENDA legislation and web accessibility guidelines.

A personal achievement came in the self development; this project has equipped the author with a range of new soft skills as well as the ability to author web sites in a range of standards and languages.

Bibliography

Baral, C. (2003). Knowledge representation, reasoning, and declarative problem solving, Cambridge University Press.

Borich, G. (1996). Effective Teaching Methods. New York, USA, Macmillan.

Christian Anger, Kathrin Konczak, et al. (2002). "A Glimpse of Answer Set Programming."

Costantini, S. (2004). Proposal of a Tutorial on Answer Set Programming. EDBT.

Dawson, C. W. (1999). The essence of computing projects: a student's guide. Hemel Hempstead, Prentice Hall.

DfES (2004). Towards a unified e-learning strategy- Consultation Document. Department for Education and Skills**:** 51-53.

Disabled Rights Commission (2003). Formal Investigation Report: Web Accessibility.

Dix, A., J. Finlay, et al. (1997). Human Computer Interaction. Harrow, England, Prentice Hall.

Duggleby, J. (2000). How to be an online tutor. Hampshire, Gower.

Exley, K. and R. Dennick (2004). Giving a lecture – from presenting to teaching. London, Falmer Press.

Given, B. (1996). "Learning Styles –A synthesised model." Journal of Accelerated Learning and Teaching **1**: 21.

Gronlund, N. E. (1991). Constructing Achievement Tests. New Jersey, USA, Prentice Hall.

Honey, P. and A. Mumford (1982). Manual of Learning Styles. Berkshire, Maidenhead.

Jeffs, T. and M. Smith (1996). "Informal education: conversation, democracy and learning."

Kolb, D. (1984). Experiential learning: experience as the source of learning and development. New Jersey, Englewood Cliffs.

Muijs, D. and D. Reynolds (2001). Effective Teaching – Evidence and Practice. London, Paul Chapman.

Nielsen, J. and R. Molich (1990). Heuristic evaluation of user interfaces. Conference on Human Factors in Computing Systems, Seattle, Washington, United States, ACM Press.

Nielson, J. (1989). The matters that really matter for hypertext usability. Conference on Hypertext and Hypermedia, Pittsburgh, Pennsylvania, United States, ACM Press.

Preece, Rogers, et al. (2002). Interaction Design – beyond human computer interaction. New York, USA, John Wiley & Sons.

Ryan, S., B. Scott, et al. (2000). The virtual university: the Internet and resource-based learning. London, Kogan Page.

S Ryan, B Scott, et al. (2000). The virtual university: the Internet and resource-based learning. London, Kogan Page.

Schneiderman, B. (1998). Designing the User Interface. Reading MA, Addison-Wesley.

Sebesta, R. W. (1996). Concepts of programming languages. London, England, Addison Wesley.

Sommerville, I. (2002). Software Engineering. London, Addison-Wesley.

Sonwalkar, N. (2002) A new methodology for evaluation: the pedagogical rating of online courses. Syllabus Magazine **Volume**, DOI:

Stephenson, J. (2001). Teaching and Learning Online – Pedagogies for new technologies. London, Kogan Page**:** 41-42.

University of Bath. (2002). "Learning and Teaching Strategy." from http://www.bath.ac.uk/quality-support/lts2002.htm.

University of Bath. (2005). "Quality Assurance Manual." Retrieved 14/05/2005, 2005, from http://internal.bath.ac.uk/quality/.

University of London Birbeck College, D. o. B. (2004). BioMoo - Principles of Protein Structure.

Whiteside, J., J. Bennett, et al. (1988). "Usability engineering: Our experience and evolution." Handbook of Human-Computer Interaction **1**(1): 791-817.

Appendices

A. Functionality / Usability Tests

The following scenarios relate to the requirements document sections 3 through 7.

Question board tests

Without logging into the tutorial, go though every page and ensure that there is no possible route that allows you to add edit or delete any of the information stored on the system regarding questions asked by other students or responses posted by tutor.

Log on to the site as a tutor. Find a question with no responses then answer the question. Ensure there is no way you can post a question.

Log on to the system as a student. Add a question that is blank. Ensure that the system informs you it was blank. Add a real question. Go to the question board. Ensure it is visible. Log out. Ensure you can still see it. Ensure the date that it was posted is correct.

Registration Tests

Register with the system starting with these details: Username - CorneliusBoriosDavrosLaVos, Password – Pips and Email Address thisisfake@fakefake.ir. Ensure that the system makes you change all three inputs.

You want to be a tutor how do you apply?

Log in as an administrator. Add a new tutor.

Login Tests

Log in to the system. Use an incorrect e-mail. Request a reminder of your Password. First type an incorrect email address. Now type your real email address. Ensure the email arrives.

Content and Assessment Tests

Log out. Find out what horn clauses are. Read the rest of the presentation. This should take around 15mins. Perform a test. Ensure this is not possible. Log in. Now perform the test. Ensure your score is visible.

Log in as a tutor. Add the tutorial found on the desktop to the system. At the bottom of the document there is a set of questions add these as an evaluation. This should take about 20 minutes.

Accessibility Tests

The following tests were suggested by Jim Thatcher who is an Accessibility consultant base in Texas. They are fairly comprehensive.

- Use your browser. Turn off images and see if what is display is understandable. Stop using the mouse to test. Make sure that your page can be used with a keyboard only. Change the fonts and colors.
- Use an evaluation tool such as Bobby.
- Use a text-only browser like Lynx, or a talking browser like Home Page Reader.

B. Test Results

Functionality Tests

Question Board

Large inputs excepted

Blank inputs screened

Meets requirements

Response Board

Large inputs excepted

Blank inputs screened

Meets requirements

Login and Registration

Large inputs excepted

Blank inputs screened

Meets requirements

Destruction test highlighted that if one user were to login into the site by clicking back to the same login screen as the user before then they remain logged in as the previous user.

Course Content

Large inputs excepted

Blank inputs screened

Meets requirements

Destruction test states that it is a security risk allowing users to potentially add scripts to the website.

Assessment

Large inputs excepted

Blank inputs screened

Does not meet requirements as problem occurred with the add slashes function that was added after the add assessment was implemented. A Tutor cannot add an assessment.

Usability Tests

Task Based analysis

The tasks set in A were performed. This was prior to the aforementioned fault developing on the system. The users were able to complete all the tasks set with the minimum of effort. Break downs occurred with all tree users when back tracking through the system to get to another task. Because of this several back buttons we added at the point of breakdown. All three users were also asked to perform the following tasks while being timed. These are the mean results.

Task	*Mean Time*
Ask a 20 word question once logged in.	3 minutes
Post a 40 word reply once logged in.	5 minutes
Post another 40 word reply.	4 minutes
Find a reply to an old question.	3 minutes
Register for a password.	3 minutes
Login to system.	30 seconds
Request a password reminder.	30 seconds
View a tutorial.	20 minutes
Complete an assessment once logged in.	7 minutes

The users were then asked if the felt that in inordinate amount of time on any of the tasks. The users agreed that the amount of time spent getting to a task was insignificant when compared to the amount of time spent keying in the data.

The following goals were set in the first draft design specification document:

Did you find the online tutorial as effective as if the content was delivered in a lecture hall: Better / Worse	*<20% Worse*
Did the site do enough to assist your learning? Y/N	>75% Yes
Was it easy to work out how to use the tutorial? Y/N	>75% Yes
The second time you used the site could you remember where things were? Y/N	>70% Yes

The users rated all of these categories favourably however they were viewing a finished product of a final year dissertation and my not have felt it was an environment for constructive criticism.

Usability Guidelines

The usability guidelines were followed quite closely but there were a few that were justifiably not met:

Place navigation bar on the right due to proximity to scrollbar and linearization for screen readers: - This was justified earlier but right alignment was not favourable in user testing, the nave bar is not the main navigator the title bar is, accessibility issues are averted with the skip to main link.

Use the access key attribute on the navigation buttons to speed up interaction for experience users and give an alternative to the mouse for mobility impaired users: - This was implemented however the tags labels became unreadable.

Use a search box – The site is not big enough to warrant a search and there should be no need if learning objectives are clearly stated.

Colour code site functions and lay up elements- As above

No non standard link colours hyperlinks should be made obvious by underlining them- The links have the same link as the tutorial menu this is done for consistency and aesthetics: - There is no picture so it is accessible. This is not deemed a problem.

Always underline links don't rely on mouse-over actions: - As above

The following three guidelines were oversights.

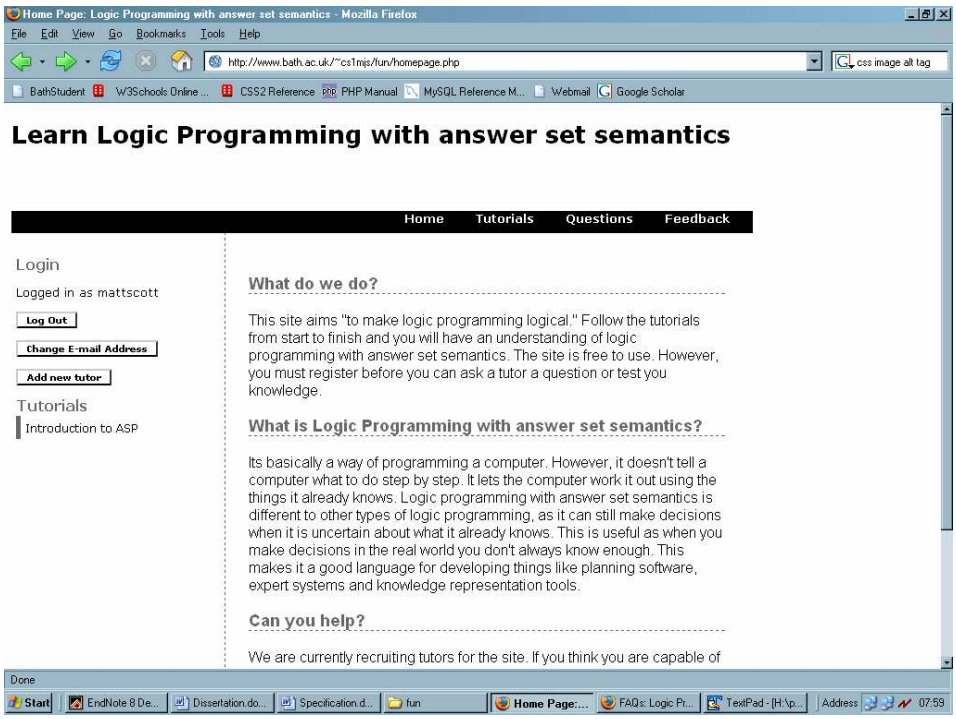
Provide printing options: - Should have been done

Include a One-Sentence Tagline: - "Making logic logical"

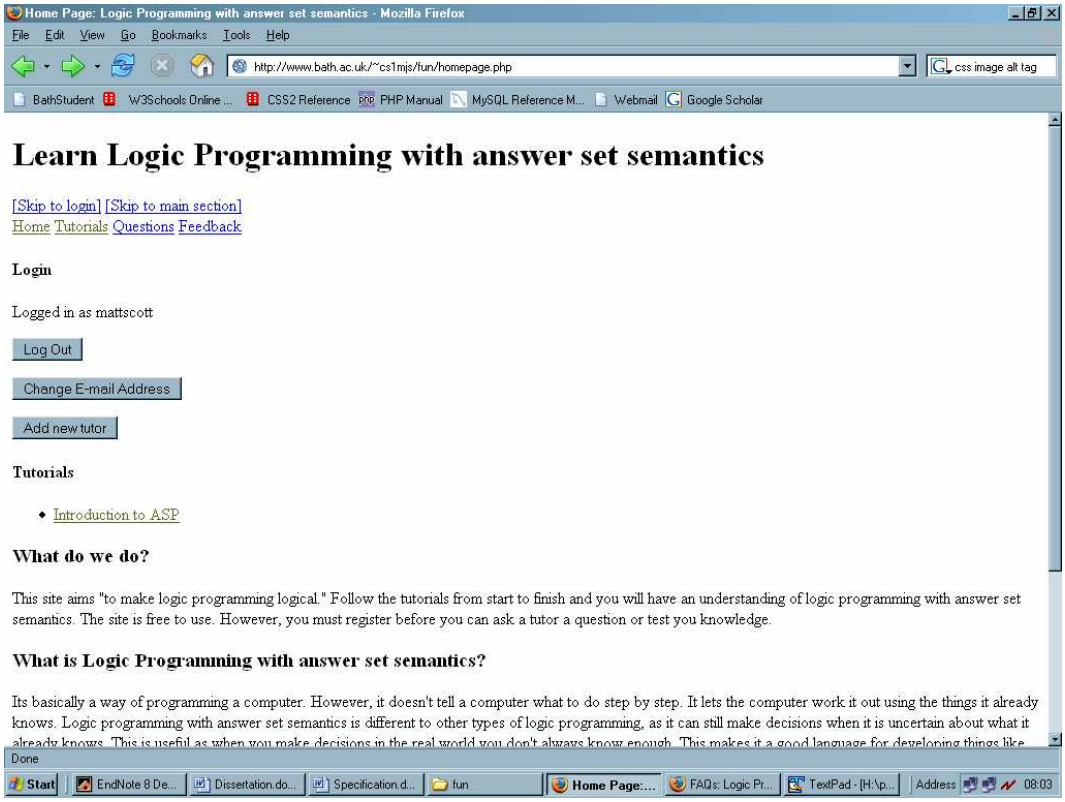
No long pages without navigation through the page: - Should have been done on tutorial page.

Accessibility tests

The tests specified by Jim Thatcher were completed accept for the download and use of the bobby validator due to technical reasons. Below is a screen shot of the site without any images:



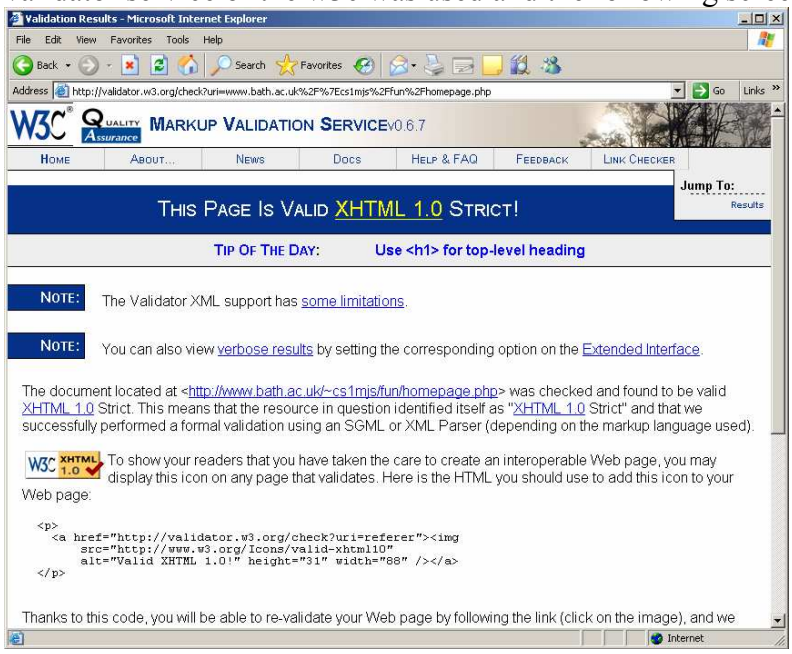
Next there is a shot of the site without a style sheet this is the closest it is possible to get to a visual depiction of the text read out by lynx.



The tab order follows the line of the page but it takes a long time to get through the menus as more and more tutorials are added it may be worth nesting the tutorials into difficulty bands.

Web standards testing

The site was predominantly validated as xhtml strict and the css was valid. The validator service of the w3c was used and the following screen was generated:



As stated in the testing section this validation service does not account for the state of the system and the entire mark-up contained within the script files. There fore it was essential to test the system on all of the browsers and operating systems mentioned in the requirements document. Apart with on glitch with displaying the title banner in safari which has been solved now, the site was compatible with all the browsers.

C. Maintenance Guide

D. Code

E. Logic Programming Tutor

Requirements Document

Introduction

Recent developments in the field of logic programming have meant that it is now possible to develop large applications using non-monotonic logic programs. Unfortunately, there is a lack of accessible information on logic programming and that which is available requires a depth of prerequisite knowledge in formal logic. If logic programming is to be pushed onto the undergraduate syllabus in a variety of disciplines then it will be necessary to produce an undergraduate introductory course on the subject.

The University's Corporate Plan articulates the aim of establishing the University at the forefront of innovation in flexible/open learning … delivered via a variety of modes, media and methods, including e-learning." (University of Bath, 2002) The creation of a software product that performs the role of a tutor in this course will not only meet a strategic aim for the University but allow the course material to be easily shared between institutions and hence support the agenda of the logic programming community.

Software Aim

To produce a software product capable of explaining undergraduate students how logic programs and their respective answer solver work.

Abstract Requirements

2. The system shall provide the following high level services to the end user:
    a. Disseminate core course content on logic programming with answer set semantics to the user.
    b. Provide a means of self assessment to the users.
    c. Facilitate interaction between users of the course and experts in the field.

User Classes and Characteristics

3. There will be the following end users of the system
    a. Students, these are casual users in a range of geographic locations who access the course in there own time. Though the students will predominantly come from a computer science background the system should still cater for novice web users.
    b. Tutors, these are experts on ASP able to answer questions raised by students. These tutors will access the system with moderate frequency and are expected to have a basic understanding of web applications.
    c. Administrators, these are responsible for the up keep of the system, maintenance of the course material, course development and the selection of tutors. They will be frequent users of the site and are expected to have an in depth knowledge of the application.

Functional Requirements

4. **Question Board**
    a. Students can post questions about the course to a course tutor.
    b. Tutors can post answers to specific questions.
    c. Users can view all questions and responses, the dates they were posted and who posted them.
    d. (a, b) require users to be logged in.

5. **Registration**
    a. Students can register their user-name and e-mail address with the system.
    b. Tutors can register with the system by sending an e-mail to the course administrator requesting a user name and password.
    c. The system will screen for fake email addresses.

6. **Login**
    a. In order to gain full access to the site a student must first register with the system then log in.
    b. The student must input their email address and password.
    c. The system will verify the username and password allow them access to the site.
    d. If the password is not verified then the student will be invited to try again or have a reminder of their password emailed to them.
    e. If the system does not recognise the user name then the student will be informed and invited to try again or re-register as a new user.

7. **Course Content**
    a. Students should be able to view a range of presentations on ASP topics stated within the course requirements section.
    b. Students do not need to be logged in the view these presentations.
    c. If a user has logged however, then the system should indicate which presentations the user has already completed.
    d. Tutors should be able to add tutorials.

8. **Assessment**
    a. After completing a presentation the students should be able to perform a short test to assess there learning.
    b. (a) Requires students to be logged in.
    c. The system should record the scores of students on these tests.
    d. These tests should be in the format of closed questions such as multiple-choice or fill in the blanks.
    e. Tutors should be able to add assessments.

Data Requirements

9. The system will be required to store the following data:
    a. User-names, passwords, e-mail addresses and user types (administrator / student / tutor).

b. Presentations users have viewed, evaluations completed and scores obtained.
c. Questions posted to the question board, the student who posted them and the date they were posted.
d. Answer to student questions, the tutor who posted the answer, the question it was in response to and the date it was posted.

Usability Requirements

10. These requirements are high level. Specific measurable goals will be stated once the software has been designed.
    a. Effective to use – The site should perform its software aim as effectively as alternative pedagogical methods.
    b. Efficient to use – Users should be able to perform within a time frame by way in which they believe it was easy and quick.
    c. Safe to use – The user should not be able to loose their stored on the system.
    d. Has good utility – The user should feel the level of functionality is suited to their use of the system.
    e. Easy to learn – The user should be able to access the full functionality of the system without external assistance.
    f. Easy to remember how to use – Upon entering the site for a second time a user will remember the layout and improve upon access times.

Accessibility Requirements

11. The system should abide by the W3C's web content accessibility guidelines < http://www.w3.org/TR/WAI-WEBCONTENT/#Guidelines>
12. The system should be dyslexia friendly abiding by the guidelines set by the British Dyslexia Association. < http://www.bda-dyslexia.org.uk/main/information/extras/x09frend.asp>
13. The content should be written in plain English and abide by the standards set by the Plain English Campaign < http://www.plainenglish.co.uk/webdesign.html>

Course Content Requirements

14. At the end of the logic programming tutorial the user will have attained the following learning objectives:
    a. Distinguish between logic programming and other programming paradigms.
    b. List strengths and weaknesses of logic programming paradigm
    c. Understand the meaning of statements in predicate calculus.
    d. Identify syntactical errors in predicate calculus statements.
    e. Outline the process of resolution
    f. Justify the use of horn clauses
    g. Explain the origins of logic programming with answer set semantics
    h. Construct simple propositional statements in AnsProlog
    i. Identify lexemes in AnsProlog

j. Comprehend simple AnsProlog programs
k. Specify the inferencing process of AnsProlog
l. Construct simple list structures and arithmetic statements in AnsProlog.
m. State the differences between AnsProlog and other declarative languages.
n. Explain how AnsProlog overcome the following problems
    i. Resolution order control
    ii. Closed World Assumption
    iii. The Negation Problem
o. State AnsProlog's limitations
p. List applications of logic programming.

Operating Environment

15. The system must be delivered via distance learning and hence will need to be provided over the internet. The software product must therefore conform to the w3c web standards. At a minimum the system must be validated at http://validator.w3.org/ and perform the functional requirements on the following we browsers:
    a. Internet Explorer 6
    b. Internet Explorer 5
    c. Fire Fox
    d. Safari
16. The system must also be able to be viewed using a range monitors with a range of colour depth. At a minimum the system must meet the functional requirement on a range of screen sizes from 800x600 to 1024x768 using a web safe colour pallet.
17. The system will also be accessed from a range of platforms and it is required to function on at least the following operating systems:
    a. The Windows family inc. Win XP, Win 2000, Win 98 and Win NT.
    b. Unix
    c. Macintosh
18. According to the World Wide Web consortium, 11% of browsers are not java script enabled and so it is important that if the system makes use of this script then it must provide an alternative format.
19. The software should be accessible to users on bandwidths above 28.8khz.

Change Log

2c. Administrators, these are responsible for the up keep of the system, maintenance of the course material and other remaining administrative duties. They will be frequent users of the site and are expected to have an in depth knowledge of the application.

Altered due to ambiguity in distinction between role administrator / tutor

18. *Added, oversight realised in specification*

6d. Added after a discussion during design with client.

7e. Added after a discussion during design with client

6a. Students should be able to view a range of multimedia presentations on ASP topics stated within the course requirements section.

Removed the word multimedia, due to PHP upload option being disabled on server hence no longer a requirement of all presentations

4c. Upon receipt of a student's details the system will send an e-mail to the user containing a password to log into the system.

Altered as on usability tests the users found this "beurocratic"

14d. Added due to web standards being misleading

F. Usability Guidelines

1. **Navigation**
    a. Use broad shallow menus
    b. Name and position menus consistently
    c. Emphasise high priority tasks by placing them higher up on the screen.
    d. Use text based navigation buttons to ease accessibility and decrease download time.
    e. Group navigation elements together on the screen.
    f. Place navigation bar on the right due to proximity to scrollbar and linearization for screen readers.
    g. Use the access key attribute on the navigation buttons to speed up interaction for experience users and give an alternative to the mouse for mobility impaired users.
    h. Provide a skip to main content link for users using screen readers to save them hearing all the navigation options each time the load a new page.
    i. Use a search box.
    j. Indicate search scope.
2. **Lay out**
    a. General
        i. Use the same page layout on every page to aid familiarisation.
        ii. No lay up used in the content pages (xhtml) this makes it easier for screen readers
        iii. Use divisions not tables as tables are used as a way of representing data not for lay up.
        iv. Align page elements to aid speed reading.
        v. Do not use frames as they don't work on mobile devices, they don't feed into search engines well, they are not printable and they don't deal with large fonts well.
        vi. Structure page meaningfully by grouping elements together by function. This helps develop the user's cognitive model of the system.
        vii. No long pages without navigation through the page.
        viii. Position using percentages not pixels to allow for text resizing and different resolutions.
    b. Colour
        i. Colour code site functions and lay up elements.

ii. Use colour wisely, design in greyscale first to ensure that users with colour deficiencies can distinguish between the colours.
iii. Have a large contrast between background and foreground colours for greater definition.
iv. Set a colour scheme using a maximum of seven colours otherwise colours start to clash.
v. Colours should exist on the 216 universal-colour palette so they are view able on all colour monitors.
vi. Don't rely on colour alone to communicate a message.

c. Dyslexia
   i. Limit lines to 60 to 70 characters. Lines that are too long or short can put strain on eyes especially with dyslexics. This against the advice of usability .gov who state that longer lines aids speed reading.
   ii. Use wide margins and headings.
   iii. Use of boxes for emphasis or to highlight important text can be effective.
   iv. Avoid dense blocks of text by using short paragraphs.
   v. Use bold to highlight. Italics, or underlining can make the words run together.
   vi. Keep lines left justified with a ragged right edge.
   vii. Use bullets or numbers rather than continuous prose.
   viii. Don't hyphenate words that are not usually split in order to fill up line ends.
   ix. The space between lines is important. Recommendations suggest a leading (space) of 1.5 to 2 times the space.

3. Links

a. Establish importance hierarchy of pages and place more important links in hot space.
b. No non standard link colours hyperlinks should be made obvious by underlining them.
c. Indicate internal and external links by including the URL below the link.
d. Begin link names with the most important keyword
e. Use text links for accessibility
f. Name and position links consistently either inline with the paragraph or on a separate line.

g. Use descriptive labels as some assistive technologies pull all the links out of content and list them together. Therefore the link needs to have meaning out of context.
h. Use Shortcuts and underline shortcut key in link name.
i. Always underline links don't rely on mouse-over actions.
j. Make tab headers look like real world tabs.
k. Show used links.
l. Allow screen readers a hidden link to skip to main content
m. Place truncation symbols between inline links so readers can easily distinguish where one line starts and another ends.

4. Web page constraints

a. Keep page sizes below 30,000 bytes to ensure a download time of less than 10 seconds
b. No complex URLs.
c. Provide a meaningful title to aid search engines.

5. Content

a. General
    i. Allow users to control time sensitive content changes.
    ii. Use content that supports site mission
    iii. Avoid blinking moving or flickering content
    iv. Provide printing options
    v. Provide a text only version
b. Text
    i. Use short paragraphs and sentences
    ii. Use adjustable font size
    iii. Where possible use lower case letters rather than capitals. Using capital letters for emphasis can make text harder to read for dyslexics.
    iv. Improve scanning with meaningful subheadings
    v. Use paging
    vi. Use familiar sans serif fonts these are easier to read especially for dyslexics.
    vii. Minimum size 10 font
    viii. Be conscious of where sentences begin on the page. Starting a new sentence at the end of a line makes it harder to follow.
    ix. Try to call the readers 'you'; imagine they are sitting opposite you and you are talking to them directly.
    x. Give instructions clearly. Avoid long sentences of explanation.

c. Tables
    i. Use the <th> tag to mark up table headers
    ii. Use the caption element and the summary attribute to describe tables
    iii. Multimedia
    iv. Use short and clear alternative text.
    v. Use alt="" for non-link images that do not convey important information or are redundant.
    vi. Or use inline descriptions or the longdesc attribute on the image.
    vii. Use a "d-link" or description link that opens the description file that is referenced in the longdesc attribute.
    viii. For client side image maps use an alt tag for each area.
    ix. For server side image maps use a redundant link to a list of links.
    x. Provide transcripts of audio and video content
d. Forms
    i. Use the title attribute on every form element and ensure labels are linked and next to the input.
    ii. Ensure you can track the focus and have a clear tab order in form elements
e. Images and multimedia
    i. Use Alt tags wherever possible
    ii. Don’t use images for decoration
    iii. Don’t use images above 72dpi as that is the screen resolution
    iv. Set the height and width of images as when a browser first loads the page they will leave a space for the image and down load it after the text.
    v. Provide a transcript for videos and audio files.

6. Branding

a. Use a logo to make users aware of where they are on your site
b. Include a One-Sentence Tagline
c. Write a Window Title with Good Visibility in Search Engines and Bookmark Lists
d. Group all Corporate Information in One Distinct Area

Usability testing