MOOClm: Learner Modelling for MOOCs

Ronald Cook

A thesis submitted in fulfilment of the requirements for the degree of Master of Philosophy

School of Information Technology
Faculty of Engineering
University of Sydney
2017
Acknowledgements

Judy Kay was the primary supervisor for this thesis (and also for my 1991 Honours thesis on a related topic). Her advice and guidance were invaluable. Bob Kummerfeld ably assisted Judy as secondary supervisor. Richard Gluga was most helpful in providing access to the ProGoSS sources and template database, and with some issues in getting it all up and running. Andrew Lum provided the Vlum source files for cross-evaluation. John Stavrakakis provided the COMP2129 2014 exam paper for evaluation with regard to Learning Objectives. Several members of the CHAI research group at the University of Sydney provided valuable advice and feedback on portions of the user interface for the final OLM interface.

Any inaccuracies in the text are entirely my own.

Summary of Publications

The material described in Chapters 3-5 was described in part in “MOOCIm: User Modelling for MOOCs,” published by the International Conference on User Modeling, Adaptation and Personalization (UMAP 2015). [Cook et al., 2015]
Abstract

Massively Open Online Learning systems, or MOOCs, generate enormous quantities of learning data. Analysis of this data has considerable potential benefits for learners, educators, teaching administrators and educational researchers. How to realise this potential is still an open question.

This thesis explores use of such data to create a rich Open Learner Model (OLM). The OLM is designed to take account of the restrictions and goals of lifelong learner model usage. Towards this end, we structure the learner model around a standard curriculum-based ontology. Since such a learner model may be very large, we integrate a visualisation based on a highly scalable circular treemap representation. The visualisation allows the student to either drill down further into increasingly detailed views of the learner model, or filter the model down to a smaller, selected subset. We introduce the notion of a set of Reference learner models, such as an ideal student, a typical student, or a selected set of learning objectives within the curriculum. Introducing these provides a foundation for a learner to make a meaningful evaluation of their own model by comparing against a reference model.

To validate the work, we created MOOClm to implement this framework, then used this in the context of a Small Private Online Course (SPOC) run at the University of Sydney. We also report a qualitative usability study to gain insights into the ways a learner can make use of the OLM.

Our contribution is the design and validation of MOOClm, a framework that harnesses MOOC data to create a learner model with an OLM interface for student and educator usage.
## Contents

1 Introduction ........................................ 8
   1.1 MOOClm’s Approach .............................. 9
   1.2 The need to integrate MOOCs with OLMs ........ 10

2 Background ........................................ 15
   2.1 Learner Models and Ontologies .................. 15
       2.1.1 Ontologies .................................. 15
       2.1.2 Learner Models ............................... 16
       2.1.3 Scrutable User Models and Open Learner Models 18
       2.1.4 Learner Model Visualisations ................. 20
       2.1.5 Ontology Visualisation as a Guide to Learner Model Visualisations 26
       2.1.6 Overview of Visualisations .................. 31
   2.2 Computer Science Education ..................... 32
       2.2.1 Knowledge Levels and the Bloom Taxonomy .... 32
       2.2.2 Pedagogies: Structuring Learning .............. 34
       2.2.3 Course Design Software and ProGoSS ........... 35
       2.2.4 The ACM CS2013 Model Curriculum ............. 37
   2.3 Massively Open Online Courses .................. 40
       2.3.1 Types of MOOC ............................... 41
       2.3.2 Problems with MOOC Participation .............. 42
       2.3.3 Motivation ................................. 44
       2.3.4 MOOC Assessment ............................. 45
       2.3.5 Recent Developments .......................... 45
   2.4 Learning Analytics and Educational Data Mining . 46
       2.4.1 Overview of Learning Analytics ................. 46
       2.4.2 Application of Learning Analytics .............. 47
       2.4.3 Learning Analytics and Learner Models ........ 48
       2.4.4 Overview of Educational Data Mining .......... 49

3 Conceptual Model .................................. 50
   3.1 Overview ........................................ 50
       3.1.1 Course and Curriculum Design and the MOOC .... 52
       3.1.2 Evidence Processing .......................... 52
3.1.3 Learner Model Design for the OLM ............................. 52
3.1.4 Key Interface Elements for the OLM ........................... 53
3.2 Course and Curriculum ............................................. 54
3.3 Population of the Learner Model ................................. 54
3.4 The Open Learner Model ............................................. 55
  3.4.1 Learner Model .................................................. 55
  3.4.2 Reference Models ............................................... 56
  3.4.3 Filters .......................................................... 57
3.5 Review of Aims ...................................................... 59

4 Design of the OLM 61
  4.1 Representation of the Learner Model ............................ 61
  4.2 Design of the Full Learner Model Ontology ..................... 62
  4.3 Design of the Visualisation ..................................... 64
  4.4 Reference Models and the Learner Model ....................... 65
  4.5 MOOClm .......................................................... 66
    4.5.1 OLM Visualisation ........................................... 67
  4.6 MOOClm: Mapping and the Reference Model .................... 70
  4.7 Resolver Tuning .................................................. 72
  4.8 Simple vs. Expert Interfaces ................................... 73
  4.9 Component Selection and Display ................................ 74
  4.10 Learning Resources ............................................. 78
  4.11 Directive to New Learning ..................................... 79

5 Implementation 80
  5.1 System Architecture ............................................. 80
  5.2 Constructing and Populating the OLM ......................... 81
    5.2.1 Parsing Evidence Logs ..................................... 82
    5.2.2 Parsing Evidence ........................................... 83
    5.2.3 Construction of Reference Models ........................ 83
    5.2.4 Construction of Filters ................................... 84
  5.3 Implementing the Visualisation ................................ 85
    5.3.1 Core Interface .............................................. 85
    5.3.2 OLM Backend .............................................. 85
    5.3.3 Internal Learner Model Representation .................... 86
    5.3.4 Visualisations ............................................. 87
    5.3.5 Privacy and State Tracking ................................ 89
  5.4 Personis Backend and Resolvers ............................... 89
  5.5 Integration with the MOOC ..................................... 90
  5.6 Statistical Analysis ........................................... 92
    5.6.1 Origin of the Data Set .................................... 92
    5.6.2 Description of the Data Set ................................. 93
    5.6.3 Hypothesis: Activity in SPOC results in improved Learning .................................................. 95
    5.6.4 Hypothesis: Assessment in SPOC indicates improved Learning .................................................. 97
5.6.5 Hypothesis: Correlation is higher when variables have a larger number of possible values
5.6.6 Other possible predictive hypotheses
5.6.7 Breaking data down by Decile
5.6.8 Statistical Result for Personis Resolver

6 Evaluation

6.1 Study Design
6.2 Study Structure
6.3 Thinkaloud Study - Outline of Participants
6.4 Thinkaloud Study - Detail of each Question
6.5 Outline of Results
6.6 Discussion concerning questions with no tied hypothesis
6.7 Conclusions regarding Individual Hypotheses
6.8 Conclusions from the Study

7 Conclusion

References

Appendices

A Nqv (New QuickView)
B Exam Learning Objectives
C Characteristics of Full Learner Model
D Restricted MOOC Ontology (Phase 1)
E Think-Aloud Study: Script
F Think-Aloud Study: Ethics Approval
G Think-Aloud Study: Personal Information Statement
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>MOOClm Core Elements</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>qv</td>
<td>21</td>
</tr>
<tr>
<td>2.2</td>
<td>LOSUM/SIV Sample Model View</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Fraction Helper - Student View</td>
<td>23</td>
</tr>
<tr>
<td>2.4</td>
<td>Fraction Helper - Parent view</td>
<td>24</td>
</tr>
<tr>
<td>2.5</td>
<td>Mastery Grids. The top summarises relative progress of student vs. group, where the bottom section shows a ranked display for all learners.</td>
<td>25</td>
</tr>
<tr>
<td>2.6</td>
<td>INGRID tree and sunburst representations of a user model. The tree representation, at top, shows the expansion of one subtree at each level and the truth values of those items. The sunburst shows all elements, but lacks space to show information on each element.</td>
<td>27</td>
</tr>
<tr>
<td>2.7</td>
<td>Sample view of the Visual Thesaurus</td>
<td>28</td>
</tr>
<tr>
<td>2.8</td>
<td>Sample Icicle plot</td>
<td>29</td>
</tr>
<tr>
<td>2.9</td>
<td>Sample of a Hyperbolic Tree (Figure 1 from [Lamping et al., 1995])</td>
<td>29</td>
</tr>
<tr>
<td>2.10</td>
<td>Sample of a Voronoi Treemap (Figure 10 from [Balzer et al., 2005])</td>
<td>30</td>
</tr>
<tr>
<td>2.11</td>
<td>Sample of a Rectangular Treemap (here from a disk usage analysis using WinDirStat). Larger blocks represent larger files and directories.</td>
<td>31</td>
</tr>
<tr>
<td>2.12</td>
<td>Sample of a Circular Treemap (from MOOClm)</td>
<td>32</td>
</tr>
<tr>
<td>2.13</td>
<td>NSW Department of Education Science and Technology Learning Outcomes</td>
<td>35</td>
</tr>
<tr>
<td>2.14</td>
<td>Moodle Learning Outcomes - Course Summary</td>
<td>36</td>
</tr>
<tr>
<td>2.15</td>
<td>List of Assessment Questions for an exam as represented in ProGoSS</td>
<td>37</td>
</tr>
<tr>
<td>2.16</td>
<td>Assessed Topics for a question in ProGoSS, including topic names, descriptions, Bloom levels and the confidence level from which a learning outcome may be concluded</td>
<td>38</td>
</tr>
<tr>
<td>2.17</td>
<td>ProGoSS Assessed Outcomes Example</td>
<td>39</td>
</tr>
<tr>
<td>2.18</td>
<td>ELLI Learning Profile spider diagram (Spider &amp; Crick, 2012)</td>
<td>48</td>
</tr>
</tbody>
</table>
6.7 List of Learning Resources displayed for Q1.5 .......................... 115
6.8 Comparison of Simple (left) and Expert interfaces .................. 117
6.9 Sample Solution: Question 1.7. Arrows indicate where changes
must be made to satisfy this exercise. ................................. 118
6.10 Question 1.9 First Visualisation: Circular Treemap with slanted
text ............................................................................. 120
6.11 Question 1.9 Second Visualisation: Sunflower ....................... 120
6.12 Question 1.9 Third Visualisation: Dendrogram Tree ............... 121
6.13 Sample Solution: Question 2.1 ........................................ 122
6.14 Sample Solution: Question 2.2 ........................................ 123
6.15 Sample Solution: Question 2.3 ........................................ 124
6.16 Sample Solution: Question 2.4 ........................................ 125
6.17 Sample Solution: Question 2.5 ........................................ 126
6.18 Sample Solution: Question 2.6 ........................................ 127
A.1 nqv ................................................................. 150
List of Tables

2.1 Summary of Visualisations ........................................... 33
3.1 Portions of Conceptual Model for each Requirement ........... 60
5.1 Modelling exam results per question vs video and problem flags . 94
5.2 Modelling exam results vs problems attempted and problems correct ........................................... 97
5.3 Matrix of coefficients of correlation ................................. 99
6.1 Thesis Hypotheses ....................................................... 106
6.2 Hypotheses Tested by each question ............................... 107
6.3 Information about participants ..................................... 108
6.4 Ease of understanding - Question 1.1 ............................... 108
6.5 Ease of understanding - Question 1.2 ............................... 109
6.6 Ease of understanding - Question 1.3 ............................... 111
6.7 Ease of understanding - Question 1.4 ............................... 112
6.8 Ease of understanding - Question 1.5 ............................... 114
6.9 Ease of understanding - Question 1.6 ............................... 116
6.10 Ease of understanding - Question 1.7 .............................. 117
6.11 Ease of understanding - Question 1.8 .............................. 118
6.12 Ease of understanding - Question 1.9 .............................. 119
6.13 Ease of understanding - Question 1.10 ............................ 121
6.14 Ease of understanding - Question 2.1 ............................ 122
6.15 Ease of understanding - Question 2.2 ............................ 123
6.16 Ease of understanding - Question 2.3 ............................ 124
6.17 Ease of understanding - Question 2.4 ............................ 125
6.18 Ease of understanding - Question 2.5 ............................ 125
6.19 Ease of understanding - Question 2.6 ............................ 126
6.20 Summary of Results Across all Questions ....................... 128
6.21 Time taken by Thinkaloud Participants .......................... 129
6.22 Time taken by hypothesis ........................................... 130
6.23 Free comments ......................................................... 134
Chapter 1

Introduction

Massively Open Online Courses (MOOCs) represent a number of new challenges and opportunities over traditional online and face-to-face courses.

This thesis demonstrates an approach to using the mass of data available from a MOOC to build a learner model, “a model of the knowledge, difficulties and misconceptions of the individual” [Bull, 2004], and make it available to the learner.

Our approach facilitates the large, long-term learner models typical of those which life-long usage is anticipated to require by basing the learner model on a broad-based ontology while permitting the student to focus on specific areas of interest.

Classes on the scale of a typical MOOC, in the thousands or tens of thousands, do not permit students to be monitored on an individual basis. This necessitates a low level of intervention on a per-student basis and introduces a risk that possibly serious individual student problems may be lost in a mass of largely irrelevant data.

We use an Open Learner Model (OLM), a learner model “that can be viewed or accessed in some way by the learner, or by other users (e.g. teachers, peers, parents)” [Bull and Kay, 2010]. This grants greater transparency to the system’s tracking of learner progress and opens the option of sharing this progress between multiple MOOCs or SPOCs (Small Private Online Courses) [Fox, 2013].

We integrate a SPOC developed using the OpenEdX platform with an Open Learner Model (OLM) built from a standard curriculum, with an ontology based on the ACM CS2013 Undergraduate Computer Science curriculum [ACM Joint Task Force, 2013].

Adding an interface to the Learner Model makes it an Open Learner Model. We assembled an interface based on a circular treemap visualisation as introduced by Kai Wetzel, here as implemented by Bostock’s D3 library [Bostock et al., 2011].

The potential to share an Open Learner Model between MOOCs suggests repeated use of the Learner Model over the long term, a Lifelong Learner Model [Kay, 2008]. Such Lifelong Learner Models are intended for use on an ongoing
and perhaps indefinite basis, integrating data harnessed across multiple applications. In contrast, many learner models studied in the literature concern interaction with single applications, with a correspondingly limited set of learning outcomes that must be represented. We discuss the design constraints for a Lifelong Learner model which differentiate it from learner models scaled for a single course or application.

1.1 MOOClm’s Approach

We attempt to integrate several core components into our framework. Figure 1.1 shows how these elements interact at the highest level.

Figure 1.1: MOOClm Core Elements

First Element: The Curriculum

The first component is the Curriculum, being the body of knowledge represented by our course. The curriculum structure may be considered as an ontology, a “set of concepts and categories in a subject area or domain that shows their properties and the relations between them.” (OED) Our background chapter discusses this in more detail.
Our Curriculum is the material which we want our students to learn. At its simplest level, it can be expressed as a set of *Learning Objectives*, or capabilities which we wish the student to learn by the end of the course, along with the level of sophistication at which we want the student to be able to accomplish each task. These “levels” represent the student’s depth of understanding. The Bloom taxonomy [Bloom and Krathwohl, 1956] is perhaps the best known taxonomy of learning levels, representing knowledge from its most basic (Knowledge) to most advanced (Evaluation).

**Second Element: The MOOC**

The second component is the actual *MOOC* - the Massively Open Online Course which teaches our curriculum.

In the terms used by OpenEdX, our MOOC platform, this is the “course-ware”. It consists of lecture videos and slides, short- and long-answer self-test questions, forums for student interaction, and any other components desired by the course designer.

These components of the course are normally referred to as *Learning Objects*, course elements constructed to assist the student in meeting the Learning Objectives. These may consist of tutorial videos, sample exercises, discussion forums, or anything else intended to assist the student’s learning. These may also be referred to as *Learning Resources*.

The MOOC may also have its own *Assessment Instruments* to test and evaluate the capabilities of the student, which typically contribute towards a final score for the course. These include (but may not be limited to) class assignments, intermediate tests and the final course exam. Assessment Instruments are the means by which educators typically determine whether a student has achieved the course’s Learning Objectives.

While building the OLM we can observer student interactions with both Learning Objects and Assessment Instruments to infer the student’s knowledge of each topic.

**Third Element: The OLM**

The third component is the *Open Learner Model*, the data store which holds information about a student’s characteristics and activity in a fashion facilitating interaction and the interface which allows the learner to interact with it.

Our Learner Model is modelled after our desired curriculum, in our case the ACM CS2013 Undergraduate Computer Science curriculum.

**1.2 The need to integrate MOOCs with OLMS**

There are several advantages to linking an OLM to a MOOC.
• Integration of the OLM permits the MOOC to **personalise content**, beyond that which is possible solely from the resources of the MOOC itself.

• The MOOC may **take advantage of learning data from other MOOCs** or from other sources, such as traditional coursework.

• **The educator** is given the opportunity to view the specific progress of the learner. If the OLM includes data from other resources, they may be informed of student capabilities that are not reflected in the MOOC.

• The OLM allows the learner to view (and possibly compare) their learning progress in a consistent fashion across multiple resources. The OLM provides an **overview** which may be absent from the MOOC itself.

With regard to tracking of student progress, studies indicate [Wilkowski et al., 2014] that students do not always approach a MOOC in the same manner as a traditional course. Rather than proceeding through a course sequentially from beginning to end, students may select individual components of the course, using the course as a learning resource, picking and choosing in a manner more common with reference works.

This creates new challenges for course and curriculum designers, who must allow for students working through nominally more advanced material without the explicit grounding of earlier parts of the course. Course designers must also allow for cases where the student has *legitimately* skipped over previous content in order to reach the material for which they have most immediate need.

As we intend that the learner model be used as a basis for lifelong learning, it follows that the same learner model should be reusable across multiple MOOCs. A learner model that integrates the knowledge from multiple MOOCs in such a fashion may be of arbitrarily large size. This is because it must be able to store all learning objectives covered by all MOOCs in which the student participates. While our implementation and interface study only concerns a single SPOC, such re-usability is vital if the OLM is to be useful in a lifelong learning context.

Designing for such a learner model introduces requirements absent from those for a particular course. The Learner Model must be adaptable to multiple course frameworks, while any interface must permit the learner to restrict their view of the OLM to whichever subset is of most immediate interest.

In an effort to address these issues, we present **MOOClm**. It represents a generalisable framework for mapping an Open Learner Model onto a course’s curriculum, in terms of learning objectives, as well as its teaching and assessment components. The framework was designed to compare student performance against a competing student or a particular portion of a *curriculum*, but also has other applications in assessing the coverage of the MOOC and its assessment instruments.
Design Goals for the OLM and the Learner Model

Our goal is to assist the student in knowing their progress in terms of learning the total content of a course, for example so that they could skip content in which they have no interest. This also permits the student to track their progress as the course continues.

Thus we design our framework to answer the following questions, as listed in our 2015 paper [Cook et al., 2015]. These concern “What” the student knows and can see.

1. **(H1) Overview:** What is the overall progress of this student on the learning activities?

2. **(H2) On-track:** In which learning objectives has the learner met the teacher expectations?

3. **(H3) Behind:** In which learning objectives are they lagging behind expectations?

4. **(H4) Activity-Type-Progress:** What are the answers to Q2-3 for a particular class of activity (video, exercise, discussions...)?

5. **(H5) Act:** How can the student find learning resources associated with any given learning outcome?

Broadly, the goal is to use a Learner Model to track the student’s progress. As we cannot guarantee that automated inferences will always be correct, our design goal is that the student should be able to make corrections to any inferences made by the system. Our Learner Model is already required to be open (i.e. an Open Learner model) by virtue of being viewable. In making it modifiable, it must also be open to user feedback.

Because we would like our learner model to be re-usable in the long term as a lifelong learner model, it should support arbitrary expansion of the OLM to include additional courses and fields, potentially representing multiple fields of a person’s knowledge. This introduces an additional dimension of scalability; we allow not only for large numbers of students, but for large numbers of topics.

We supplement our core five questions with several design goals chosen to further use of our OLM for lifelong learning - or “How” we structure our OLM and its interface specifically for lifelong learning.

6. **(H6) Openness:** To permit student interaction for the Learner model, it needs to be not simply a learner model, but an Open Learner Model. The need for student corrections to the learner model requires that the OLM be modifiable as well as simply viewable.

7. **(H7) Selectivity:** There must be a way to view parts of their model selectively, so that large learner models are not unusably unwieldy.
8. **(H8) Topology Mapping:** To match different possible arrangements of the course material, the topology of the OLM as presented to the student must be modifiable. For example, remapping the course topology would allow the student to view material with learning objectives organised by week or by topic, or to group topics with their logical prerequisites.

9. **(H9) Comparison:** To permit comparisons of a Learner Model against other students - and against selected components of the curriculum - we would like to permit comparison of Learner Models. The “students” may be notional (“ideal” or “typical”) or real. This is in some ways a more general form of our *Activity-Type-Progress* question.

10. **(H10) Curriculum:** Since we are comparing one set of learning objects (what the learner knows) with another set of learning objectives (the objectives covered by a curriculum or a learning object) we should be able to use our toolset to cross-compare these selected “slices” of the curriculum.

We designate these goals as H1-H10. These represent ten hypotheses that we believe our design successfully accomplishes.

The interface should permit ready overview of the model as a whole, as well as selected parts of the OLM, in a helpful manner - it should facilitate learning and introspection.

In addition, we attempt to design our OLM in such a manner that it is well-suited for lifelong learning. Basing the OLM on a standard curriculum makes it easier to map the OLM against new MOOCs using the same curriculum. Since the student may participate in many MOOCs, we must also allow for the student’s complete learner model to include the learning objectives of all such MOOCs. While the full Learner Model may be very large, the student is likely to only be interested, at any given time, in that portion of the OLM concerning their current course or learning focus. As such, we need to allow for the student to see a selected slice of a potentially enormous OLM. The topology mapping tools for H8 will be useful for this purpose. Since we only examine a single SPOC, we cannot test whether we have met these goals successfully. As such, they are not listed amongst our hypotheses.

**Outline**

Chapter 2 reviews background work on Learner Models, Curriculum Design and different approaches to visualising ontologies.

Our approach is outlined in Chapters 3-5. Chapter 3 describes the overarching Conceptual Model. Chapter 4 then describes design of our OLM, including the underlying ontology and the visualisation. Chapter 5 outlines the implementation in additional detail.

In Chapter 6 we demonstrate that our framework successfully answers our core questions and design goals. This is done by a set of worked examples and by a small-scale thinkaloud study.
Chapter 7 concludes.

Our contribution is in, firstly, the creation of an Open Learner Model suitable for use with MOOCs in a lifelong learner context; and secondly, the creation of the MOOClm framework for integrating this OLM with a MOOC. Key innovations of this framework are the use of filters to limit visible scope, and the use of a “Reference Model” to explicitly compare learner models.
Chapter 2

Background

Our contribution integrates components from several different fields to mutual advantage. The primary fields involved are Learner Modeling (or User Modeling), Computer Science Education, and Massively Open Online Courses. We also discuss the field of Learner Analytics, required to build a learner model from mass data.

2.1 Learner Models and Ontologies

Learner Models are a specialised case of the broader area of User Modeling. A User Model is a “the system’s set of beliefs about the user’s knowledge, preferences, goals and attributes” [Kay, 1999]. In many respects, the core of a user model comprises a user-specific mapping of a field of knowledge against an ontology intended to represent the knowledge and beliefs of a user.

2.1.1 Ontologies

An ontology “defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members).” [Gruber et al., 2009] It may be viewed as a computer-based specification for a “knowledge map”, a computer model of a field of knowledge. In this respect, any organised collection of knowledge may be used to seed an ontology with concepts and information. However, an ontology in the Information Technology sense of the term specifically refers to a representation of knowledge held within a computer.

An ontology may be lightweight or heavyweight. A lightweight ontology is essentially a list of concepts, knowledge and beliefs, with little representation of how the represented information inter-reacts. A heavyweight ontology, on the other hand, adds information about how the components of the ontology interrelate, “adding axioms and constraints to lightweight ontologies” [Corcho
et al., 2003]. In a lightweight ontology, there may be knowledge components for “addition” and “subtraction.” A heavyweight ontology would have these components and additional information indicating, for example, that if $a + b = c$ then $a = c - b$ (relating addition and subtraction algebraically) or that subtraction is equivalent to the addition of a negative number.

2.1.2 Learner Models

This thesis uses the definitions referenced in section 2.1 for Learner Models and User Models. Where a User Model is a representation of a user’s knowledge, beliefs and characteristics, a Learner Model is a User Model intended for use in an educational context. We will be using both terms depending on the applicability of the concepts under discussion. User Models have been used for over twenty years as a tool to track learner progress in achieving educational course outcomes [Kay, 1994]

User Models and Ontologies

In many respects a User Model may be seen as a user-specific instance of an ontology. Where an ontology represents what may be known about a field of interest, a User Model represents what is known or believed by a particular user. As such, it is possible to take an ontology and convert it into a form suitable for use as a user model. However, user models may extend the ontology by adding user characteristics that are absent in the baseline ontology.

Examples of information which may be represented in a User Model which do not correspond to elements of an ontology include:

- User-specific characteristics such as name, age and gender.
- Information about the person’s learning style and preferences.
- Information about educational background and other courses completed.

All of these items carry some obvious use in an education context. The person’s name can be used to personalise presentation or to draw attention. Age may be used to draw examples from material with which a user is more likely to be familiar. Gender may be used to adjust names and genders of participants in examples in order to encourage sympathetic interaction. Information about learning style may be used to adjust the ordering and presentation of material.

Another key concept to be integrated when building Learner Models is information about depth of knowledge. A learner model must represent not only what is known, but how well it is known and the degree to which the learner is familiar with a concept. The most common taxonomy used for this is that proposed by Bloom et al. in 1956 [Bloom and Krathwohl, 1956], which was subsequently revisited by Krathwohl with many revisions. [Anderson et al., 2001, Krathwohl, 2002] These changes included the reversal of the two highest “levels” and extension with additional elements such as creativity. However, where
relevant, this thesis limits itself to the earlier work due to its existing extensive use in the literature. This is discussed in additional detail in section 2.2.

It is possible to simplify representation of a user model based on a heavy-weight ontology into a lightweight form, while losing some internal semantic information. In the example given earlier relating the concepts of addition and subtraction, the relationship explicitly linking addition and subtraction becomes a new knowledge component, for example, “knows that subtraction is equivalent to adding the negative of the same number.” The semantic link between the concepts is then imposed from without by the model’s interpretation of available evidence.

As mentioned in our Introduction, an Open Learner model (OLM) is a Learner Model “that can be viewed or accessed in some way by the learner, or by other users (e.g. teachers, peers, parents)” [Bull and Kay, 2010]. We discuss this further in section 2.1.3.

**Building the User Model: Evidence**

In building the user model, it is necessary to have some source of evidence from which to decide the state of each model component. Generally this has been done via an explicit link to the software from which evidence is collected. Mapping from arbitrary evidence in an arbitrary format to a standardised user model format presents a significant challenge. The publication of the TinCan API [Kelly and Thorn, 2013] provides a standardised format for storage of evidence and so acts as a translation layer for collection of evidence between arbitrary software packages and the user model. Another approach is suggested by Veeramachaneni et al., who break down the most common interactions with a MOOC and design their MOOCdb database schema around this data [Veeramachaneni et al., 2013]. The approach typified by MOOCdb bears the weakness that anything outside of the projected interactions cannot be represented; for example, there is no model for collaboration between different MOOCs.

However, data on learner knowledge as tracked formally by the MOOC may not always be available. It may be necessary to extrapolate user patterns not from data provided directly by a MOOC or LMS, but from the raw web server logs maintained by the system. Zaïane et al. described how raw server logs could be used to infer additional data about the learner [Zaïane and Luo, 2001, Zaïane, 2002]. One problem with this approach is that most standard tools for server log analysis are aimed at commercial interests in order to extract purchasing data rather than student progress, necessitating the use of customised tools or, at minimum, tight tuning of the commercial packages.

Regardless of intermediate representation of the evidence used, it is often necessary to use Learner Analytics or Educational Data Mining techniques to actually extrapolate from available evidence to valuation of models. What patterns in viewing videos represent active exploration of the subject material? When answering assessment questions, does answering a question successfully after several incorrect answers indicate actual knowledge of the topic at hand, or has the learner simply been picking answers at random until finding the
right one? What depth of knowledge is possible when nominal prerequisites are not satisfied? It is such interpolations and extrapolations that require use of analytical techniques.

**Reusability**

Kay’s 1999 paper [Kay, 1999] also discusses the utility of reusability in a user model. Once a user model is available, it can potentially be used not only in that context but in others. A Learner Model associated with teaching basic cryptography techniques may also reference a different part of the model concerning number theory in order to determine whether prerequisite mathematics is known.

Some Learner Model designs have been incorporating reusability as a basic feature [Brusilovsky et al., 2005, Kay and Kummerfeld, 2012] in part as a step towards lifelong modelling [Kay, 2008]. This approach also reduces the need for rebuilding possibly complex OLMs and OLM agents for different courses [Chen and Mizoguchi, 2004], especially given the developing potential for reuse of courseware [Stewart et al., 2005, Fletcher et al., 2007].

For a user model to be re-usable, it must be held in a format that is recognisable by multiple systems. One approach is typified by GUMO [Heckmann et al., 2005] which builds on the standardised userML markup language to represent the model.

The Personis server [Kay et al., 2002] provides a secure framework for storage and access to the user model, including facilities for limiting access to particular parts of the model and protocols for remote access.

Personis also provides for use of distinct user model “Resolvers”, a selection of alternative internal code functions that calculate the truth value of a model component by giving different weighting to different evidence depending on user or application preferences. For example, one resolver may ignore video evidence; another may ignore evidence after a particular point in time in order to find the value at that time; or may choose to ignore explicit evidence to examine how a component resolves independent of user input.

Personis has a strong emphasis on user scrutability of their user model. Resolved values are chosen not at the point when the model is populated, but by a resolver which is supplied when values for model components are queried. Models may be accessed locally or remotely and incorporate permissions on a per-application and per-user basis for each model.

**2.1.3 Scrutable User Models and Open Learner Models**

Once the decision is made to create a user model, one must decide how widely the model will be available. The default with most software is to hide the model from the user; when running Google searches, they do not reveal to you the database they use in tuning your search results.

Scrutability of user models (that is, permitting the user to scrutinise their own model) has been shown [Cook and Kay, 1994, Lum, 2007] to improve stu-
dent understanding and recognition of areas where effort is required for additional learning. Use of user models for educational feedback was explored as early as the early 90s [Cook and Kay, 1994], with more recent studies demonstrating much more sophisticated visualisation tools, permitting users to gain greater insight into their progress towards educational objectives. Lum [Lum, 2007] examines the case for scrutability in user models to permit learners to examine those areas where the system “believes” their progress in course completion to be deficient.

This research has contributed to the concept of the “Open Learner Model” (OLM), a learner model available both to the student and to the software being used for their education.

Bull and Kay describe in their SMILI paper [Bull and Kay, 2007] some of the dimensions involved in determining whether a student model should be viewable and how it should be presented, covering the many visualisations used for presenting student model data to the learner. SMILI presents a framework in which to view the extent and purpose of a user model and its scrutability.

The SMILI framework has four parts:

1. **Context and evaluation**: How does the open learner model fit into the overall interaction and how was it evaluated? Is evidence interpreted consistently depending on who is viewing it? (For example, a teacher may choose to view the model in a way which ignores student claims about things that they know that are not supported by direct evidence.) How important to the purpose of the model is its degree of openness? Is the model presented in a fashion that is easy to understand? How is the model used?

2. **WHAT is open?** Is the entirety of the model visible? If not, which parts are hidden from which viewers or applications, and what is the basis for these decisions? Is raw evidence available, or only final conclusions? Who is able to make changes?

3. **HOW is it presented?** How is the model presented to the student? Are parts available but hidden in the default view? Are multiple views available? How are different components highlighted? Is the view tailored differently for the student compared to the teacher?

4. **WHO controls access?** Can the user block access to others? Can the user permit access to others? Is access controlled by the student, or by the applications that constructed the student model in the first place? How is data protected?

These four components can be used to evaluate the extent to which a student model is “open” or merely appears to be. The SMILI paper was written to summarise several previous published papers in terms of how the user models exposed fit into the SMILI framework.

The SMILI framework was revised in a 2016 paper [Bull and Kay, 2016] in the context of OLM developments since the original paper’s publication. The
newer paper suggests small modifications of the framework in the light of more recent developments.

One interesting possibility arising from use of Learner Management Systems and MOOCs is that a student may choose to compare their user model with other students. Hsiao et al. found [Hsiao et al., 2011] that allowing students to compare the progress of their learner model against that of other students resulted in a substantial (39%) increase in the average number of attempts at each quiz question compared to a control group where the parallel visualisation was not available. However, availability of parallel views also led to more advanced students making 25% fewer attempts, which is unlikely to be a desirable outcome. There is also some risk to students who make poor progress; 23% of students in this survey indicated that they would only wish to show positive progress to their peers.

In 2006, Bull and Mabbott [Bull and Mabbott, 2006] also explored the idea of comparing an Open Learner Model against their peer group and against instructor expectations for their progress in the course using a skill meter visualisation. Eighteen of twenty-three respondents agreed that a comparison to instructor expectations was useful (with five neutral responses). Twenty out of twenty-three agree that a peer comparison was useful (with two neutral and one disagreement.)

2.1.4 Learner Model Visualisations

Once a user model is created and populated, in order for it to actually be scrutable, some form of representation is required. At its simplest level this can be a simple text interface listing the knowledge components and values. However, scrutability is advantageous for learner feedback and tuning of their learner model [Kay, 2000]. It follows that the representation of the model for the user should be designed towards those purposes.

The learner model should be scrutable and permit learner interaction; easily understood; permit comparison against different learner models; be scalable for large learner models; and transparent in function.

An “ideal” model viewer would enable all of these, but in practice any given design will favour some of these tasks over others. Andrew Lum’s “LOSIV” viewer [Lum, 2007] highlights those areas of most interest, permits evidence to be examined and allows uninteresting “clutter” to be hidden, but will only show a single user’s model at any time.

In contrast, Loboda et al.’s “Mastery Grids” [Loboda et al., 2014] present a very simple representation of learner progress while giving a quite nuanced representation of learner progress compared to the student’s classmates.

We examine here a selection of visualisations from the literature and one visualisation created for this project. These are summarised by Table 2.1 on page 33.
qv

Figure 2.1 illustrates qv (QuickView), a simple visualisation designed as an early experiment in presenting a learner model to the student along with the evidence chosen to populate it [Cook and Kay, 1994]. The model was presented as a hierarchical tree; each node could be clicked to collapse or expand all available sub-nodes. Any resolved node could be right-clicked to bring up a list of associated evidence.

Qv is straightforward but not particularly scalable, and lacks information on the detail of each element of the learner model. It also lacks references to learning resources.

LOSUM/SIV

SIV, illustrated in Figure 2.2, is a visualisation designed by Andrew Lum [Lum, 2007] to highlight those items of most interest and so guide the student to the most relevant next topic. While in the default view, low-level topics are shown jumbled together and unreadable, the interface includes provisions to filter out much of the overlapping material and so enable the viewer to focus more easily
on key parts of the model. The viewer includes a detailed view of the evidence used to populate each model component when selected; the figure shows an example of this in the large right pane.

**Fraction Helper**

Fraction Helper [Lee and Bull, 2008], shown in Figure 2.3 and Figure 2.4, is a simple OLM-based application designed to assist students in learning how to manipulate fractions correctly, and in teaching their parents how to correctly instruct their children. As such, the information modelled is quite limited in domain, but is presented in two distinct ways. The representation shown to the student (Figure 2.3) includes pictures of healthy or “sad” trees, depending on what misconceptions are held, along with text telling the student where their problems probably lie. The visualisation for the educator (Figure 2.4) focuses on specific concepts and where improvement is most needed.

More recently, Aleven et al. [Aleven et al., 2016] use a similar approach for their “Example-based tutor”, using a variety of different interfaces.

**Mastery Grids**

Mastery Grids [Loboda et al., 2014] are shown in Figure 2.5 These were implemented as a means for the student to track their own progress through a
Figure 2.3: Fraction Helper - Student View
Figure 2.4: Fraction Helper - Parent view
Figure 2.5: Mastery Grids. The top summarises relative progress of student vs. group, where the bottom section shows a ranked display for all learners.

course and compare their progress against other students as well as a model “standardised progress” pseudo-user representing where the student’s progress should lie in the course material. Emphasis is heavily on social comparison, with little effort made to demonstrate underlying knowledge structure or supporting evidence. Most students found availability of the model to be useful and results showed that many students were spurred on by seeing their progress compared to that of others. One possible issue of concern is that the most advanced students appeared to be somewhat demotivated rather than motivated by the comparison.

Ioana Jivet’s Masters Thesis [Jivet, 2003] simplifies the approach used by Mastery Grids by summarising the comparison of the student’s progress via a simple spider chart representation comparing the student’s activities to that of the “Average graduate.” Metrics such as the number of videos watched, timeliness and accuracy of quiz submissions and total time on the MOOC platform are displayed for comparison. These details are not broken down by topic, showing only overall MOOC activity levels for the student. The results of her study showed a higher rate of engagement in and graduation from the MOOC, but
she could not conclude that learners improved self-regulated learning skills as a result.

**INGRID**

The INGRID system [Conejo et al., 2012] provides a public web service for presentation of user models. User models may be handed off in XML form, after which they are processed to present a visualisation in one of two forms. The first form is a simple tree structure, with only a single branch (plus its “ancestors” and ancestor siblings) displayed at any time. The second format uses a radial sunburst representation. In both forms, colouration of the represented nodes indicates how well known the items on that branch are known. Samples of these, from the paper, are shown in Figure 2.6.

The API used for access to the web service includes hooks for passing data (including change information) back to a contact point supplied in the configuration XML data, as well as other data which should permit presentation of evidence if properly used.

Each of the two representations has its own strengths and limitations. The “tree” diagram can only show a single path (and siblings at each level), which can make navigating the full tree difficult; essentially, the visibility it provides is quite limited in scope. In comparison, the “sunburst” representation gives an effective overview, but the amount of data in the user model rapidly becomes unwieldy.

### 2.1.5 Ontology Visualisation as a Guide to Learner Model Visualisations

While many different OLM visualisations have been proposed and implemented in the literature, we can also draw on a much richer literature if we take full advantage of the conceptual link between ontologies and learner models.

In the earliest history of knowledge representation, ontologies were proposed by Aristotle in his classification of the natural world, extended into a logical tree structure by Porphyry and visually represented in the High Middle Ages. Linnaeus organised life into a logical tree when establishing his taxonomy of species, an idea which was used by Darwin in *The Origin of Species* [Darwin, 1859] including a diagram representing the branching of species through a tree.

However, we are more interested in more recent work in computer-based visualisations of ontological structures.

These include tree and sunburst representations as demonstrated in INGRID, but many other visualisations have been proposed and implemented. Many ontology visualisations are designed for a fundamentally static view; others may be used interactively, but have not been used for representation of learner models.

For example, we may look at the “force graph” as used in the online Visual Thesaurus [Thinkmap, 1998-2015] and for which implementations are available in several public libraries including the D3 library [Bostock et al., 2011] used in
Figure 2.6: INGRID tree and sunburst representations of a user model. The tree representation, at top, shows the expansion of one subtree at each level and the truth values of those items. The sunburst shows all elements, but lacks space to show information on each element.
our own implementation. Force Graphs are visually attractive in part due to their dynamism, but this very dynamism makes the representation of a particular structure unstable, making it difficult for the learner to locate and track particular elements. Figure 2.7 demonstrates this with the word “make”, while also demonstrating why the representation is impractical for larger structures.

Another visualisation used is the “icicle” representation, developed by Kruskal et al. in the early 1980s [Kruskal and Landwehr, 1983] where all items are shown hierarchically as rectangles with size proportional to the significance of the the object represented. Figure 2.8 shows a full-tree representation using this method, again using Bostok’s D3 library; this example is sourced from the examples on the D3 web site.

Another readily available visualisation is the hyperbolic tree representation as first designed at Xerox PARC [Lamping et al., 1995]. The example in Figure 2.9 shows Figure 1 from the original Xerox paper. One may zoom into any particular node which focuses on that part of the tree.

Its most notable problem is the degree to which fine-grained structure is minimised due to being restricted to the diagram’s margins. This is an issue with many visualisations; the smallest components are fundamentally arranged in a single dimension. That dimension may be horizontal or vertical, or along the circumference of a circle, but fundamentally, one dimension is reserved for representation of structure, while another attempts to represent the smallest
Figure 2.8: Sample Icicle plot

Figure 2.9: Sample of a Hyperbolic Tree (Figure 1 from [Lamping et al., 1995])
However, there are some visualisations which make more extensive use of internal space. The Voronoi treemap, “based on a mathematical segmentation of Euclidean space that dates back to Descartes” [Lima, 2014] was proposed as the basis for visualisation of software metrics by Balzer et al. in 2005 [Balzer et al., 2005] and later re-purposed to represent ontological hierarchies. Figure 2.10 reproduces Figure 10 from Balzer’s original paper. This structure makes highly efficient use of available space, but in so doing the actual structure being represented is somewhat obfuscated.

A simpler form of treemap is based on rectangular divisions. While similar layouts had been used previously, the modern approach is usually ascribed to Schneiderman et al. [Shneiderman, 1992] with later extensive refinements. Figure 2.11 shows an example of this layout in representing disk usage in the freeware program WinDirStat.

This shares the problem of the Voronoi treemap: while space is used very efficiently, structure is obfuscated by the dense packing used.

Finally we examine the “Circular Treemap.” It appears that while these were first implemented by Kai Wetzel in an open source implementation, they were
first referenced academically in a review paper concerning the state of the art in treemaps by Schneiderman [Shneiderman and Plaisant, 1998].

As noted by the original author and by Schneiderman’s review paper, circular treemaps are somewhat wasteful of space; as also noted, however, they make structure clearly visible. However, space usage for representing N nodes is still $O(\sqrt{N})$, using internal space much more efficiently than representations with a linear dimension such as simple trees.

Because this format combines relatively efficient use of space (compared to $O(N)$ representations) with clear representation of structure, it was selected as our baseline visualisation for MOOClm. A sample is shown in Figure 2.12.

Readers interested in the field of visualisation of ontologies are referred to Manuel Lima’s *The Book of Trees* [Lima, 2014].

Grundy et al. recently reviewed a number of different Open Learner Model visualisations in support of Formative Assessment [Law et al., 2015]. However, while their paper did cover a wide range of alternative visualisations, they limited their analysis to very small-scale models, with only three outcomes shown for most examples. These approaches are of limited utility in examining large open learner models.

### 2.1.6 Overview of Visualisations

Table 2.1 summarises some of the visualisations covered here and their strengths and shortcomings in terms of those dimensions which are important for this thesis.

Here “Scalable” represents whether the visualisation can display very large learner models (greater than, for example, a hundred entries) effectively. “Compare Models” reflects whether the visualisation permits the user to compare two
different learner models. “Focus on Learning” indicates whether the visualisation provides useful guidance to the student on future learning opportunities.

This overview suggests that a Treemap format with supplemental features to permit comparison of learner models and refer students to learning resources may be helpful. This is because the format is scalable while preserving the model structure. Its shortcomings may be overcome by careful adaptation of the visualisation.

2.2 Computer Science Education

We briefly consider whether the use of computers in education is actually helpful. Several reviews of the literature [Kulik et al., 1980, Kulik and Kulik, 1991] as well as more recent individual studies [Shakibaei et al., 2011] have concluded that learning outcomes are more reliably achieved when computers are included when assembling learning solutions.

2.2.1 Knowledge Levels and the Bloom Taxonomy

A key concepts in education is one of taxonomies of knowledge levels. These differentiate degrees of familiarity with a concept. The most widely used taxonomy is the Bloom taxonomy [Bloom and Krathwohl, 1956]. Originally outlined in 1956, it covered 6 terms, from simplest to most complex and from concrete to abstract:

- **Knowledge** - Recall or rote learning
<table>
<thead>
<tr>
<th>Name &amp;</th>
<th>Scalable Learner Model?</th>
<th>Compare</th>
<th>Focus on learning?</th>
</tr>
</thead>
<tbody>
<tr>
<td>qv</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LOSUM/SIV</td>
<td>Somewhat</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Fraction Helper</td>
<td>No</td>
<td>No</td>
<td>Excellent</td>
</tr>
<tr>
<td>Mastery Grids</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INGRID</td>
<td>Somewhat</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Force Graph</td>
<td>Good, but chaotic and hard to track individual elements</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Icicle</td>
<td>Medium - linear scaling when expanded</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rectangular Treemap</td>
<td>Good, but structure nonobvious</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Voronoi Treemap</td>
<td>Good, but structure nonobvious and size/shape of elements is inconsistent</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Circular Treemap</td>
<td>Good, but some wasted space</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Grundy et al. (various)</td>
<td>Poor</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of Visualisations

- **Comprehension** - Grasp the meaning of a concept
- **Application** - Being able to use the concept in new situations, without being prompted
- **Analysis** - Able to break down a problem or communication and identify component parts and their relationships.
- **Synthesis** - The ability to use the concept to create a new assembly which solves a nontrivial problem.
- **Evaluation** - Criticism, commentary; the ability to recognise the strengths and weaknesses of a work incorporating the concept.

Each term encapsulates the terms lower in the hierarchy, so understanding at an “Analysis” level is assumed to include understanding at the Application, Comprehension and Knowledge levels as well.

The taxonomy was revised in 2001 [Anderson et al., 2001] and an additional dimension (the “Knowledge Dimension”) added to the existing scale (the “Cognitive Dimension”). The original noun terms were modified to a roughly equivalent scale of verbs: Remember, Understand, Apply, Analyse, Evaluate and Create. It is worth noting that the equivalencies for the last two terms have been exchanged, so “Creat[ion]” (equivalent to Synthesis) is in the new taxonomy regarded as more complex than “Evaluate” or Evaluation. One of the editors later released a much shorter, simpler summary of the changes made [Krathwohl, 2002].

However, despite these revisions, the original Bloom taxonomy remains in much wider use than the revised taxonomy, perhaps due to its familiarity and
widespread use, or perhaps due to confusion from the changes made, particularly the exchange of the two highest levels.

Other taxonomies abound. For example, the neo-Piagetian taxonomy uses only three levels (Pre-Operational, Concrete Operational and Formal Operational) [Lister, 2011] and Ackoff [Ackoff, 2010] describes four (Data, Information, Knowledge, Wisdom). Note Ackoff uses “Knowledge” as the second-highest level whereas under Bloom it is the lowest; the same term is used to label different depths of concept.

These taxonomies are significant in Computer Science Education because they can be used to indicate the depth at which students are expected to know the course material. Each component in the course requirements can indicate not only the basic concept used (such as iteration), but the depth of understanding with which the student can be expected to use that knowledge.

2.2.2 Pedagogies: Structuring Learning

There is more to the facilitation of learning than a simple knowledge base.

One of the key characteristics of education is the pedagogy used. Pedagogy is “the method and practice of teaching, especially as an academic subject or theoretical concept” (OED) If an ontology is the “what”, the pedagogy is the “how.”

Siemens outlines the theory and cognitive basis for much of how the structure of learning is established in the modern, networked environment [Siemens and Baker, 2012]. He describes three core epistemologies and adds a fourth:

1. Behaviourism, which asserts that learning is a “black box” activity, in that we do not know what occurs inside the learner, focuses its efforts on managing external, observable behaviours, and finds much of its existence in objectivism.

2. Cognitivism, which spans a continuum from learning as information processing (a computer model) at one end, to learning as reasoning and thinking on the other, finds much of its identity in pragmatism.

3. Constructivism, which covers a broad spectrum of research overlapping with cognitivism, contends that learning involves each individual learner making sense and constructing knowledge within his or her own context; it finds its foundation in interpretivism.

To the three-fold view of epistemology, Stephen Downes (2006) adds a fourth ... Connectivism posits that knowledge is distributed across networks and the act of learning is largely one of forming a diverse network of connections and recognizing attendant patterns.

Of these, the last three form the basis of the most common approaches used by Learning Management systems:

1. Cognitivism forms the basis of the instructivist approach, driven by the largely one-way transfer of knowledge from teacher to student.
2. Constructivist education focuses on “learning by doing,” with creation being the primary driver for learning.

3. Connectivist education make a knowledge base available then encourages students to learn from the knowledge base in collaboration with each other, often with teachers available as an additional resource for information outside of the provided knowledge base.

In practice, most Learning Management Systems use a combination of methods. While the EdX MOOC is nominally instructivist in approach, for example, it includes a forum discussion board system to facilitate connectivist learning.

### 2.2.3 Course Design Software and ProGoSS

Part of the evolution of the educational process in recent years has been the increasing formalisation of learning outcomes. It is now commonplace for a curriculum to include a list of desired learning outcomes. For example, if one seeks to discover the learning outcomes for the NSW Department of Education Science and Technology curriculum, they can be found quite easily [NSW Department of Education, 2014] as in Figure 2.13.

However, while modern courses and their assessments are typically modeled quite closely on a curriculum and its listed learning outcomes, feedback from
the assessment process to the course designer is often minimal. Assessments are modeled after learning outcomes, but the mechanisms in place to determine whether those learning outcomes are being met by the course may be absent.

Essentially, what is needed is a feedback mechanism to provide assurance that assessment results actually indicate that learning objectives are met. We wish to avoid those cases where knowledge is either taught or assessed incorrectly, then that fault missed because the failure is lost against the background of successful results.

Linking learner outcomes to learner management systems is not a new idea. Vandepitte at al [Vandepitte et al., 2003] describe the process of linking a Learner Course Management System (Ariadne) with a Learner Management system (Blackboard). The Moodle LMS, also used as a MOOC, tracks learning outcomes as well, but while it does track assessment results, it does not permit assessment to specify the knowledge (Bloom) level at which a question was answered. Each outcome is graded as to how well the course cohort knows that outcome, but a single-number, linear scale is used. Figure 2.14 shows an example from Moodle’s demonstration site.

Richard Gluga’s PhD thesis [Gluga, 2013] and a number of foreshadowing papers [Gluga et al., 2012, 2013] describe ProGoSS, which tracks Program Goal progeSSsion of a course. This is done, firstly, by setting up a number of learning goals and levels (for example, learning the concept of iteration at the Bloom “Understanding” level), then by indicating which parts of the different assessment exercises in a course which parts of the knowledge base. Different taxonomies may be selected for each Learning Outcome, so the course designer may choose to assess learning in terms of using the Bloom, revised Bloom, neo-Piagetian or any other selected taxonomy. Figure 2.15 shows the assessment form for the final exam for one course. Figure 2.16 shows how learning
outcomes are specified for a particular question.

In so doing, ProGoSS establishes a formalism which tests the effectiveness of a course in teaching its material at a “bare pass” and “top performer” level, with each learning outcome assessed not only in terms of gross success level, but also at the depth with which each outcome was learnt. Figure 2.17 shows ProGoSS’ summary of selected outcomes across a class.

ProGoSS also includes a module to test and advise assessor expertise in determining the Bloom (or neo-Piagetian) level at which an assessment exercise is aimed. In so doing, inaccuracies in testing course progression may also be reduced in the manner in which they are defined as well as in how they are assessed. However, this capability is not of primary interest for our purposes. What is useful in ProGoSS for our purposes is the ability to link learning outcomes and types to course elements. This mapping is potentially useful in establishing the learning outcomes for elements of a MOOC course.

2.2.4 The ACM CS2013 Model Curriculum

Many online courses teach fundamentally the same things, but present material in a different order. This makes cross-comparison of student objectives between courses more difficult. The obvious response to this is to map the course ontology (the material taught by the course) to a common reference framework.
Figure 2.16: Assessed Topics for a question in ProGoSS, including topic names, descriptions, Bloom levels and the confidence level from which a learning outcome may be concluded.
Figure 2.17: ProGoSS Assessed Outcomes Example
Many formalised courses already do this as a first step towards professional certification.

One of the basic ideas followed in our implementation is to build our ontology on the basis of a common reference framework so that all courses can be represented with a common, widely accepted set of learning outcomes. For our purposes we use the recent final publication revision of the ACM CS2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science [ACM Joint Task Force, 2013]. However, it rapidly became clear when attempting to map this against the course being used for comparison that the ACM curriculum contains a number of (probably deliberate) blind spots. In particular, the curriculum is almost entirely language- and environment-agnostic. A learning outcome will address the concept of iteration and looping, but does not explicitly tie this to the for or while loop in C. The curriculum will discuss processes and threads, without referring to how these are handled under UNIX or Windows. Our approach to addressing these issues is discussed in the body of this thesis.

The ACM curriculum does not use Bloom or Piaget in representing knowledge levels, although it is loosely based around Bloom, “in part because several Bloom levels are driven by pedagogic context... in part because we intend the mastery levels to be indicative and not to impose theoretical constraint on users of this document.” Instead the ACM uses three levels of “mastery.” The first is Familiarity, understanding of a concept and its meaning. The second is Usage, where “The student is able to use or apply a concept in a concrete way.” The third and final level is Assessment, the ability “to consider a concept from multiple viewpoints and/or justify the selection of a particular approach to solve a problem.” (Ibid, page 34)

The mismatch in representation of mastery levels (for example, between Bloom and ACM representations) may introduce issues with cross-comparability of learning outcomes. If a course designer has designed the course learner outcomes around the Bloom taxonomy, determining whether those course objectives - as mapped in the ACM representation - corresponding to those learner outcomes has been met is made correspondingly more difficult. There is no clear, one-to-one, bidirectional mapping between the two taxonomies.

2.3 Massively Open Online Courses

Massively Open Online Courses, or MOOCs, are a relatively recent development. “A MOOC is an online course with the option of free and open registration, a publicly shared curriculum, and open - ended outcomes.” [McAuley et al., 2010] McAuley goes on to describe the general characteristics of a MOOC:

An online phenomenon gathering momentum over the past two years or so, a MOOC integrates the connectivity of social networking, the facilitation of an acknowledged expert in a field of study, and a collection of freely accessible online resources. Perhaps most importantly, however, a MOOC builds on the active engagement
of several hundred to several thousand “students” who self-organize their participation according to learning goals, prior knowledge and skills, and common interests. Although it may share in some of the conventions of an ordinary course, such as a predefined timeline and weekly topics for consideration, a MOOC generally carries no fees, no prerequisites other than Internet access and interest, no pre-defined expectations for participation, and no formal accreditation. (Ibid, page 4)

MOOCs present formal course materials to any individuals who want to use them, to encourage learning outside of a formalised environment. Because they are designed to support large cohorts of students, their design supports evaluation performed either by automated systems or by peer group.

MOOCs are an evolution of earlier automated tutoring systems that were frequently based on presenting material to much smaller class groups, known at the time as Asynchronous Learning Networks (ALNs) [Hiltz, 1998]. These persist today in the form of Learner Management Systems such as Moodle and Blackboard. The more modern systems typically include additional refinements to encourage the learning process by social participation and increasing the learner’s stake in the course by mechanisms such as peer assessment. MOOCs themselves are a fairly recent phenomenon, only appearing under that name since around 2007.

2.3.1 Types of MOOC

Siemens [Siemens, 2013] groups MOOCs into three categories. These are xMOOCs, cMOOCs and quasi-MOOCs. Lisa Lane [Lane, 2012] suggests addition of another category, for “Task Based” or constructionist MOOCs. The differences between these largely concern their choice of pedagogy.

The first category is that of cMOOCs, such as Downes’ and Siemens’ CCK08 [Siemens and Downes, 2008]. “cMOOCs are based on a connectivist pedagogical model that views knowledge as a networked state and learning as the process of generating those networks and adding and pruning connections.” [Siemens, 2013]/cMOOCs emphasise networking and student collaboration as a pathway to learning. Students are not simply co-learners, but are expected to be co-teachers, assisting and mentoring each other and exploring the knowledge space of the course in order to gain a deeper understanding of the material. These MOOCs strongly encourage student collaboration in solution of exercises and coverage of material.

Rather than being centred on course material, a cMOOC is centred on the communications channels of the student. This approach is especially well suited to courses using peer assessment for student exercises.

The second type of MOOC is the xMOOC, such as most edX MOOCs [Breslow et al., 2013] and Coursera, which follow a more traditional structure of “teacher as expert” and “learner as knowledge consumer” [Siemens, 2013]. A typical “xMOOC” such as edX [Breslow et al., 2013] places copies of lectures and
course material on site, usually (but not always) with lectures split into smaller pieces, and typically with small assessment exercises interleaved between tutorial material. Social functions such as internal forums are included to encourage student interaction, but the pedagogy is at its core a fairly traditional “instructivist” [Lipson, 2013] one-directional affair, with the student expected to absorb material without themselves being part of the teaching process. While xMOOCs are currently the best-known and most widely publicised type of MOOC, they also make less use of the format’s advantages than do cMOOCs.

By their nature as a body of knowledge intended for transfer, xMOOCs are well suited to representation of content in a Learner Model.

Siemens’ third category is that of “quasi-MOOCs” which provide material as a loose assembly of tutorials without format structure as a course. He specifies the Khan Academy [Thompson, 2011] as an example of this format. Where an xMOOC assembles a large amount of material into a common structure, a quasi-MOOC does not impose any particular structure to learning. It is this failure to impose a course structure that leads to Siemens’ “quasi-” designation.

Lisa Lane [Lane, 2012] includes xMOOCs and quasi-MOOCs in a common category and adds her own third category of “task-based” or constructivist MOOCs, which focus on performing tasks and demonstrating skills in a practical manner. She cites the DS106 “online course on digital storytelling” run by the University of Mary Washington as an example of this final approach. Emphasis is not on passive absorption, nor on social collaboration, but on “learning by doing”. These have a strong social element, but the emphasis is not on networked learning but on performing tasks to construct a solution. Pedagogy on these is a mix of constructivism and instructivism.

2.3.2 Problems with MOOC Participation

Most current MOOCs suffer from issues with ongoing student engagement; the number of students who complete the course is typically a small proportion of those who started it. Completion rates in the single percentage range are not uncommon [Khalil and Ebner, 2014].

While completion rates in computer-driven Learner Management Systems (including MOOCs) are frequently low, formal college-level computer science courses also have relatively high dropout rates [Baubouef and Mason, 2005]. The difference is in scale; where a formal face-to-face course might lose 50% of its students between its first year and its second, attrition rates well over 90% are normal in the typical MOOC [Rivard, 2013].

A literature review by Sunar, Abdullah et al. [Sunar et al., 2015] examines much of the literature on personalisation in MOOCs. The paper “demonstrate[s] that there is a growing trend of researchers embarking in the possibility of implementing personalisation and adaptation in MOOCs in order to improve users’ engagements, hence reduce MOOCs’ drop-out rate problem.”

Here we discuss some of the reasons proposed for this dropoff.
Course awareness and the “funnel of participation”

Clow discusses the dropoff in participation in some detail [Clow, 2013], drawing a parallel to marketing concepts, suggesting that dropoff rates are due to increased commitment at each stage in the “funnel of participation.” Of the number of people who become aware of a course, relatively few visit the course’s web site; of those, fewer still register; of those, fewer again complete a significant portion of the course material; and finally, quite a small number actually complete the final assessment.

In contrast, Kizilcec et al. [Kizilcec et al., 2013] looks at statistics for participation in a MOOC over time and finds that students can be loosely grouped by initial and ongoing participation. The student population is broken into four broad categories:

- **Sampling** - who only view a very small portion of the course material; usually, but not always, at the start of the course.
- **Disengaging** - Students who do a small number of assessments at the beginning of the course and then either drop out completely or else only watch a small number of videos.
- **Auditing** - Students who engage by watching videos but not participating in assessments
- **Completing** - Students who attempt the majority of assessments, through to the end of the course. This is the pattern typical of a traditional formal course

In general, the proportion of students completing the course at later levels scaled inversely with the difficulty of the course. High School level courses were “Completed” by 27% of the student population, whereas graduate level courses were only completed by 5%. In addition to some differences in demographic backgrounds, students who engaged in the course forums were found to be much more strongly represented in the “Completing” cluster. This supports the intuition that social involvement is correlated with course engagement and success.

For some, their Motivation for Participation does not require Course Completion

Following this, Wilkowski et al., performed a study on student progress for a MOOC designed to teach use of the Google location APIs including Google Maps, Google Earth and MapReduce. The study shows that much of this differentiation appears to be due to differences in student motivation [Wilkowski et al., 2014]. Close to 50% of the students registering for the Google course had no intention of completing it. Some registered specifically to learn about particular subtopics; some wanted to learn without necessarily receiving the associated certification; and some registered simply to see how the course was constructed.
This suggests that a metrics for course completion may be insufficient in evaluating the success of a MOOC. Many students may leave a MOOC before it is “complete” according to standard metrics, while still having successfully achieved their own goals from participation.

Course Material - Difficulty and Quality

John Daniel [Daniel, 2012] discusses the history and problems with the current generation of MOOCs in some detail. Many of the best-known MOOCs were created by institutions that established their substantial reputations not via teaching, but from research. He contrasts these with the MOOC offered by the University of Phoenix - which bases its reputation on teaching - which achieves a completion rate over 30%.

Unfortunately Daniel does not differentiate on the basis of the difficulty of the material; one unexamined possibility is that the poor pass rates for the Ivy League MOOCs are due in part to the material presented being exceptionally difficult.

2.3.3 Motivation

One explanation for the poor completion rates for MOOCs is motivation. In a class setting, one is competing with their classmates and has plentiful opportunities to compare solutions and results. This is significantly more difficult outside of a classroom setting. Other motivations missing from a MOOC may include a financial stake in participation, the grading (or otherwise) of classes, and interference from full-time work.

More modern MOOCs typically include some measures designed to address the widely reported issues with learner engagement. Khan Academy [Thompson, 2011] includes a system of badges and awards to apply gamification principles to improve student motivation. EdX includes a complex set of socialisation tools to encourage student interaction and explicitly encourages students to use these as part of its introductory set of reference material. However, use of extrinsic motivators such as badge awards has been demonstrated not to be uniformly effective [Abramovich et al., 2013] with low-performing students being more motivated by a badge system than higher-performing students.

Use of social engagement has been consistently found to be positively correlated with course completion and learner outcomes. Alario-Hoyos et al. examine the correlation of several different methods of social engagement with the outcome of their course [Alario-Hoyos et al., 2013] and found that those students with a high level of engagement with social media such as course forums were also those students most likely to complete the course successfully. Similar results have driven MOOC deployments in a direction where social participation is highly encouraged. The question of whether students participate in such forums because they are enthused about the course, or are enthused about the course because of their forum participation, is not entirely decided.
One study demonstrated that videos used for presentations within the MOOC were likely to be more effective if kept reasonably short. Guo et al. [2014] demonstrated that selecting the “right” video length, no more than about six minutes, significantly improved student engagement. Other factors such as the speaker talking relatively rapidly and switching between a “talking head” and course material were also found to be helpful.

However, the literature in general suggests that linear, non-interactive video is largely ineffective in achieving learning outcomes [Zhang et al., 2006]. This was borne out by our own statistical analyses of data for the SPOC in our study.

One other possible motivational factor is that of assessment. Typically MOOCs are assessed automatically by computer. As such, there is no easy comparison with other students’ results or progress. Feedback mechanisms such as Loboda’s work on Mastery Grids [Loboda et al., 2014], as discussed earlier, may be helpful in rectifying this.

2.3.4 MOOC Assessment

One problem faced by MOOCs which is not uniformly seen in standard LMSes is the question of how students are to be assessed. With a potential audience in the thousands, lecturers do not have time to mark assessments for all students. There are two basic approaches most commonly followed to address this. The first is one of peer assessment: students are asked to assess the exercises submitted by their peers. Piech et al. discuss some methods Piech et al. [2013] used to improve accuracy of peer marking results (such as reassigning assessments to high-scoring students and away from “snap graders”) but found that for “54% of students, after all rounds, we are still unsure of their submission’s true score.” However, peer review has the added advantage that students are exposed to the work of other students and so may learn from the alternate exercise solutions offered. This may have synergistic effects for learning outcomes when working within a cMOOC framework, but comes at the price of accuracy of assessment, and so of the acceptability of such courses for formal certification.

The other method used is automated grading systems [Balfour, 2013]. While these are obviously effective for simple-form answers such as multiple-choice, as answers become more complex, producing a system capable of automated grading becomes more difficult. While Automated Essay Scoring (AES) software has been found to be quite reliable for shorter assessments, its efficacy for longer essays and creative material in doubt, and typically a seed group of training essays is required. Peer review will miss some elements missed by an automated grading system, but may catch elements that an AES will miss.

Automated grading in technical fields such as Computer Science [Douce et al., 2005] and Physics [Matt, 1998] may also be used by some MOOCs.

2.3.5 Recent Developments

More recently, many tertiary institutions have introduced SPOCs (Small Private Online Courses) using MOOC-based software to present course materials
to internal participants as a resource to supplement face-to-face teaching time and formalised assessments. The University of Sydney ran a small SPOC of this nature based on its COMP2129 course, “Operating Systems and Machine Principles,” which has been used as a model “MOOC” for this thesis.

Joseph Jay Williams, formerly of Stanford’s Lytics Lab, established a group looking at “MOOClets”, very small course components based around no more than one or two modules but made publicly available in the same manner as a MOOC [Williams et al., 2014]. In some respects these resemble less-structured courses such as those offered by Khan Academy; in Siemens’ MOOC taxonomy they would be quasi-MOOCs, but the reasons for implementation differ.

2.4 Learning Analytics and Educational Data Mining

Here we briefly review the fields of Learning Analytics (LAK) and Educational Data Management (EDM). We summarise the nature of LAK, its applications and its usefulness in populating Learner Models. We then touch on the nature of the EDM field.

2.4.1 Overview of Learning Analytics

According to the 1st International Conference on Learning Analytics and Knowledge, “Learning analytics is the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs” [Siemens and Baker, 2012]. Learning Analytics focuses on the data trail left by students interacting with the networks and systems that surround them.

... every click, every Tweet or Facebook status update, every social interaction, and every page read online can leave a digital footprint. Additionally, online learning, digital student records, student cards, sensors, and mobile devices now capture rich data trails and activity streams.

These learner-produced data trails provide valuable insight into what is actually happening in the learning process and suggest ways in which educators can make improvements [Siemens and Long, 2011].

MOOCs, by their nature, can produce enormous streams of data for each student. Some of this is the result of explicit assessment exercises and can be used to judge, potentially with significant accuracy, whether a student knows the answer to an assessment question. Much of it is only useful for relatively weak, inferential evidence, such as the observation that a student has played a particular part of a video.

Where forum data is available, evidence from the forum may be slim or may be significant; however, parsing such large masses of natural language data and inferring knowledge from it is a highly complex task.
Learning Analytics is sometimes abbreviated as LAK, for Learning Analytics and Knowledge.

2.4.2 Application of Learning Analytics

In the course of building learner models for each student, we are initially restricted largely to passive data collection, without explicit interaction with the student. Most of the time, the material available is limited to what we can see from their interaction with the MOOC. If the MOOC is actually a SPOC we may also be able to gain additional insight from feedback from formal evaluation instruments such as tutorials or exam results.

Because of this, we need a mechanism for analysing the event stream and assessments retrieved from the MOOC and using them to decide which components of our user model we can regard as “true” or “false.” Essentially we need to extrapolate a user’s knowledge from their actions, in the possible absence of direct evidence.

When discussing user modelling, we also briefly described the necessary process of interpreting evidence to build our learner model. Extrapolating from learner activity to learner knowledge via use of learner analytics is an obvious application of LAK methods. Kay and Lum addressed this exact problem in their 2005 paper Kay and Lum [2005] which concerned examining web logging data to populate a learner model based on their activity interacting with learning software and subsequently the learner models that had been built from them.

Ideally, a statistical approach could be used to infer the significance of individual actions in indicating whether a learning outcome has been achieved. A set of students who had learnt through the MOOC and has also performed formally assessed tasks would have their set of actions cross-correlated to decide the significance of each action. In practice, issues with statistical significance and sample size would prevent this. As most MOOCs for which formal, human assessment is performed are actually SPOCs, a sample size of a few hundred is likely. Attempting to infer the effect of several hundred categories of actions on an array of tens or hundreds of learning outcomes with such relatively small sample sizes is likely to lead to loss of statistical significance.

If we assume that the final assessment from a “genuine” MOOC is reliable, there would likely be sufficient data to infer the effectiveness of particular actions in moving towards particular learning outcomes.

One issue with making such inferences is the problem that “absence of evidence is not evidence of absence.” We have no real evidence of negative outcomes except for internally supplied incorrect answers. We can work with the conservative assumption that anything for which direct evidence is lacking is unknown, but we cannot be certain, especially when the learner has access to learning resources that we cannot monitor.
2.4.3 Learning Analytics and Learner Models

However, simple inference of knowledge is only the most obvious use of LAK principles. Shum & Crick [2012] demonstrate a use of Learner Analytics which carries significant potential utility in a Learner Model context. Rather than simply inferring what the learner knows, they attempt to infer the learner’s own proficiencies and characteristics and then make this information available to the learner in the form of a simple spider diagram (see Figure 2.18). This sort of information carries significant potential not only in adjusting what is presented to the learner, but how it is presented and in adjusting software behaviour in a meaningful manner.

Availability of this sort of user information, falling outside of the bounds of a simple ontology, presents both one of the greatest opportunities and one of the greatest challenges when constructing a learner model. In terms of opportunities, the power of having learning and personality characteristics available when tuning presentation (or, indeed, interpretation) of information may add a great deal to the usefulness of storing information as a learner model. The concerns are as great if not greater:

- How do we decide on the personality characteristics to be encoded in the user model?
- How do we decide when these might have changed?
- How do we avoid misinterpretation of terms out of context? For example, in Figure 2.18, the value for Resilience is quite low. This could easily be misread as indicating a general lack of personal conviction rather than its original, somewhat technical interpretation in a specific context.
- On what basis do we limit access to this information?
The ability to store information of this nature is one of the strongest arguments in favour of using a Learner Model rather than a simple database of information tracked and completed.

2.4.4 Overview of Educational Data Mining

Educational Data Mining (EDM) addresses many of the same problems as Learning Analytics but with a different emphasis.

“Educational data mining (also referred to as "EDM") is defined as the area of scientific inquiry centered around the development of methods for making discoveries within the unique kinds of data that come from educational settings, and using those methods to better understand students and the settings which they learn in.” [Baker et al., 2010]

The difference is primarily one of history and emphasis. LAK is centred on learners and the process of learning, whereas EDM is data-centred, an extension of the broader field of data mining.

EDM addresses progress of students as a group. This can then be used to judge whether a course is successful in achieving its overall objectives, or for classifying students within a course by their behaviour. In our discussion of MOOCs, for our purposes, the chief purpose of EDM is to establish a benchmark performance for the entire student cohort against which to judge individual student success, but as our primary focus is on the learner, EDM techniques will not be used to a significant extent.

The Kizilcec study cited previously [Kizilcec et al., 2013] is an example of Educational Data Mining, where student behaviour was analysed on a mass basis and clustered into differentiated populations.

Romero and Ventura in 2016 reviewed the literature on use of educational data science in MOOCs [Romero and Ventura, 2016]. They list the basic types of MOOC and discuss the communities that have interest in the data generated. They go on to discuss the history of the field and some of the challenges involved, not only for EDM, but for LAK and Academic Analytics.
Chapter 3

Conceptual Model

Given the state of prior art, we now aim to design a software system to solve the problem at the core of our thesis: integration of a MOOC with an OLM in a manner facilitating lifelong learning.

We aim to do this by, first, creating an OLM suitable for use with MOOCs in a lifelong learning context. Secondly, we link a MOOC with the OLM to form the MOOClm framework.

Let us first review our goals as outlined in our introduction. The hypotheses for these capabilities are outlined in additional detail in the Introduction in table 6.1.

The student should be able to review their overall progress (Overview) and where they have met teacher expectations, and failed to meet those expectations (On-track, Behind). They should be able to see these things for a particular class of activity (Activity-Type-Progress), and be able to find learning resources associated with a particular learning outcome (Act).

The student should be able to view and modify their data (Openness). They should be able to focus their view on important parts of it so that large learner models may be grasped without the student being overwhelmed (Selectivity). If alternative arrangements of the material are available, the student should be able to see the material so arranged (Topology Mapping). The student should be able to compare their progress against that of other students and parts of the curriculum (Comparison). Finally, would also be useful if our tools could also be used to compare segments of the curriculum.

With these goals in mind, we next consider the component parts of our design.

3.1 Overview

Our core components were outlined in the Introduction. These consist of the Curriculum (and its ontological structure); the MOOC platform (in our case, OpenEdX) and learning resources; the Learner Model, the data store holding
information about the user and their learning outcomes; and the Viewer, the interface and visualisation which makes these available to the student and educator. The Learner Model and Viewer together comprise the OLM.

Figure 3.1 shows how information and data passes between the conceptual components of the system.

Figure 3.1: MOOClm Integration and Dataflow

The conceptual design of the MOOC corresponds to the “Course and Curriculum Construction” box, designated (A) in this diagram. The actual MOOC and course materials are designated (B). These components, designated with the colour blue in this diagram, contain essentially no novel concepts.

The conceptual design corresponding to the OLM, including its interface elements, is limited to the box designated (C), shown in green. This comprises most of the novel elements of this conceptual model and as such represents the core contribution of this thesis.

Finally, boxes (D) and (E), in yellow, represent the processing of evidence from the MOOC into the form required by the OLM and the integration of data from other sources. These components incorporate ideas which are somewhat novel, but are largely already represented in the literature.

The remainder of this section describes in detail the design of these components. We then proceed to describe them in detail.
3.1.1 Course and Curriculum Design and the MOOC

First, we consider the role of the course and curriculum in our design. This corresponds to the box labeled (A) in Figure 3.1. For a more detailed discussion of this topic, see Section 3.2.

We start by considering the list of learning objectives defined in our standard curriculum. These are supplemented by any additional learning objectives not described in the standard curriculum, resulting in our curriculum design. This curriculum design drives the design of the course, in turn defining the nature of the materials created for the MOOC.

We use a standardised course as a baseline to improve reusability if the OLM arising from the course is later used with other courses.

If the course is designed prior to its formal curriculum, this process may be reversed by identifying the learning objectives addressed in the course materials and whether they can be found in the standard curriculum. This process may be systematised using curriculum mapping software such as ProGoSS [Gluga et al., 2012].

Once the course design has been determined, it is used to assemble the course materials for the MOOC (B). Once the course is running, any student activity is recorded in the MOOC’s activity logs. We do not discuss the general topic of MOOC design here.

3.1.2 Evidence Processing

Evidence collected from the MOOC is used to populate the Open Learner Model, which is stored using a suitable OLM server (Openness goal, (E) in figure 3.1). Each possible activity in the MOOC is mapped against a corresponding learning objective and the resulting evidence injected into the Learner Model Data Store in part (C) of figure 3.1.

This mapping process may incorporate evidence from other sources, such as formal exams or information about prior certifications ((D) in figure 3.1).

3.1.3 Learner Model Design for the OLM

The Learner Model is structured on a one-to-one basis on the curriculum design, with one Learner Outcome for each such present in the Curriculum Design.

Since the Curriculum Design may include elements present in our standard curriculum which are not found in the MOOC itself, this list of learner objectives may include learner outcomes which are absent from the MOOC. However, all learner outcomes for the MOOC should be present in the full Curriculum Design, even if this requires supplementing the baseline Standard Curriculum.

Adding learner outcomes which are present in the MOOC but absent from the Standard Curriculum does risk some lack of standardisation in the structure of the Learner Model. However, omitting them risks losing learning data specific to the MOOC. Introducing nonstandard elements leaves the option of mapping between nonstandard learner outcomes at a later time.
The learning objectives described in the curriculum are then used to design the Learner model Ontology. For this purpose any learning objectives present in the Standard Curriculum but absent from the present course are retained. This facilitates lifelong learning by ensuring that a broad sample of the learning objectives for the MOOC’s field of endeavour is already present for later re-use of the learner model.

For example, the ACM curriculum has a learning outcome for “Choose appropriate conditional and iteration constructs for a given programming task. [Assessment].” This falls under Fundamental Programming Concepts within Software Development Fundamentals. An introductory programming course would have this as a learning outcome. A course teaching the C programming language would assume this knowledge as a prerequisite and would instead have a learning outcome as knowing the details of using the for command in C. However, as the ACM curriculum does not go into detail on constructs for particular programming languages, a C programming course would have a new learning objective for the “for” command. A student demonstrating knowledge of the “for” command might then be inferred to have knowledge of the ACM Learning Outcome.

Evidence data from the MOOC and elsewhere is injected into the Learner Model. Viewing the resulting Learner Model enables the student to view their progress (Overview goal).

3.1.4 Key Interface Elements for the OLM

This resulting Learner Model store is then, optionally, compared to a Reference Model. Reference Models may be models of real or notional other students or of selected portions of the curriculum (On-track, Behind, Activity-Type-Progress and Comparison goals).

Because the learner model does not omit material from the standard curriculum, the learner model may be imponderably large. This potential issue is addressed by introducing the concept of filters.

This comparison model constructed by combining the original and reference model is passed through a filter to select a particular portion of the learner model which may be of interest. The filter constructs an alternative logical hierarchy for the learner model and maps the learning objectives of particular interest onto the resulting structure.

The resulting, typically reduced, learner model is then presented to the learner using a visualised interface - the Viewer (Selectivity, Topology Mapping goals).

This filtering capability addresses the issues presented by an extremely large learner model simply by omitting them from the remapped ontological topology.

The Viewer has access to a database of learner resources, in the form of an “ideal” Learner Model storing possible evidence for each learner outcome. This database may be used to refer the student to learning resources for any selected Learning Outcome (Act goal).

We now discuss portions of this process in additional detail.
3.2 Course and Curriculum

Ideally, the process of course design begins with the construction of a planned curriculum for the course. If one is available, a standard curriculum, such as the ACM CS2013 undergraduate curriculum, is chosen as a “seed” for the course curriculum.

The curriculum design identifies those elements of the course that are present in the standard curriculum (principally in the form of learning objectives). Any learning objectives present in the course that are absent from the model curriculum are selected to supplement the standard curriculum.

This curriculum design is then used as a baseline when designing the course itself and, in turn, all course materials for the MOOC.

It is possible, and may be necessary, for the flow of this process of deriving course design from curriculum design to be reversed. If the original curriculum design is unavailable or the course was designed without reference to a formal curriculum, the curriculum design may be unavailable. In such cases, it may be necessary to construct the curriculum from pre-existing course materials, again using the model curriculum as a template.

It may also be necessary to revisit this process if the course is revised, perhaps in the light of student response.

The conversion between curriculum design and course design should produce as a byproduct a mapping between learning objects - that is, elements of the MOOC courseware - and learning objectives. This mapping can later be used at step (E) in our structural diagram to convert between the raw evidence represented in the log and the learning objectives which are coded into our learner model.

The result of this process is a curriculum listing learning objectives from the Model Curriculum, supplemented by those added by the course designer; the formal course design; the course materials used in the MOOC; and a mapping between those materials and the curriculum’s learning objectives.

3.3 Population of the Learner Model

Final population of the learner model for an individual learner may draw upon multiple data sources.

One possible source is the MOOC’s own internal storage concerning what the student has viewed and what exercises they have completed. However, the MOOC’s logging data may well provide the same basic information, augmented with timing information and data on failed and successful attempts. As such, we favour the latter.

Actual mapping from course materials to learning objectives may make use of the mapping already produced for the curriculum mapping process.

A second source may be found in the learner’s interaction with the open learner model. Once they have access, if we provide a facility for modification
of the model, we can incorporate the learner’s own knowledge about what they have and have not learned.

While the educator may not choose to trust this data, it should certainly be taken into account. Ideally, the chosen OLM server will have a facility for variant interpretation of available evidence so that evidence may be weighted accordingly.

Finally we may choose to populate the learner’s data from external sources. In a SPOC setting, this data may be represented by results from formally evaluated assignments and exams. Any such material will need to be mapped against the curriculum’s learning objectives, as had already been done for the MOOC courseare.

Sourcing evidence from multiple sources in this fashion introduces the possibility of using learner model data to cross-compare the effectiveness of different teaching and assessment instruments, providing deeper insight into the effectiveness of the course and perhaps feeding back into the course-curriculum loop.

3.4 The Open Learner Model

The structure of the Open Learner Model and its novel interface elements comprise the core contribution of this thesis.

3.4.1 Learner Model

Once the Curriculum has been mapped to its corresponding Learning Objectives, this structure is converted into a Learner Model on a one-to-one basis, with each Learning Objective in the Curriculum represented in the Learner Model.

We base our Open Learner Model on a “Lightweight Ontology” modelled after our desired curriculum. A suitable open learner modelling framework, such as Personis or GUMO, is chosen to represent the learner model.

The Visualisation is then used for the student to interact with the Learner Model; this may include referrals to additional resources.

We design our Viewer to retrieve Learner Model data from one or two sources, and integrate this data. It then maps this into a more useful subset before presenting the learner model visually.

While this gives us a Learner Model which we can open, i.e. make into an OLM) using a suitable interface, it fails to address certain core requirements. The OLM in isolation lacks context; it would be useful to permit comparison against other students and assessment instruments. The OLM, because it incorporates the whole of a standard curriculum, is also too large; we need a way of scaling down the model to something more palatable to the student.

We address these issues by integrating two new capabilities into the interface component of our OLM. These are Reference Models and Filters.
3.4.2 Reference Models

A Reference Model is an OLM representing another student or a selected portion of a curriculum. A Reference Model may be used to compare the progress of two students, or of a student with a selected portion of a curriculum, or even to compare a student at one point in time with their performance at a different point in time.

In essence, therefore, there are two forms of the reference model. The first represents an actual or notional evidence record, such as a student or “ideal student” as proposed by Kay & Lum in 2005 [Kay and Lum, 2005].

These “User Reference Models” carry the appearance, and in some cases the actuality, of being for real users. They may be used to compare a student’s position against that of a competing student, or of an expected progress record for the student at a given point in time.

One key Reference Model is the “Ideal” Learner Model, based loosely on the “Plausibly ideal” student proposed by Andrew Lum [Kay and Lum, 2005]. This model contains one evidence item for each learning object in the MOOC. As such, it serves as a reference point for a notional “student who has done everything”; however, since all learning resources are recorded against corresponding learning objectives, it may also serve as a database of available learning resources. As such, it doubles as a resource for the student to locate learning resources for each learning objective.

The other form is the “Curriculum Reference Model”. These are stored as learner models, but the evidence does not necessarily represent actual student activity, real or notional. Instead, evidence is simply used as a positive placemarker to indicate those learning objectives that correspond to a selected portion of the curriculum. This allows the student to compare their own progress against a subsection of the curriculum which is of particular interest.

These Curriculum Reference Models may model a topic of particular interest, or the learning objectives assessed by a particular assessment instrument. Alternatively, they may represent a chosen alternative curriculum, perhaps for a particular complementary MOOC, or perhaps the set of advanced topics that are of particular interest to the educator.

Examples of such models include a model for “all topics covered by videos,” or perhaps “all topics covered in the final exam”, or most simply “all topics covered by this MOOC.” We can also define reference models limited, or perhaps excluding, those elements present in the model curriculum.

The range of possible reference models of both types is quite broad, limited only by the interest of the educator and their ability to create the reference model itself. The OLM interface itself may be used to create a reference model by manually modifying an otherwise blank OLM, so constructing a new reference model is quite straightforward.

Use of Reference Models lends considerable flexibility in how the toolset is used. It can be used as a simple view of student progress within a limited curriculum, or to compare the subsets of a larger curriculum taught by two different courses. There are other applications, in particular when used in combination
Reference Models interpret available evidence in the same manner as the Primary model. Our design used the Personis server’s support of custom “Resolvers” to tune this behaviour; this is discussed in additional detail in Chapter 4.

The representation of this comparison between the Standard and Reference Models requires the ability to visually present a combination of two independent values in a common interface. While many different visual variations were possible, we chose to broaden the standard palette of two colours (true/false) to four. These four colours represent:

- **True** - Both models have the same value, which is “true” or “known”.
- **False** - Both models have the same value, which is “false” or “unknown”.
- **Exceeds Expectations** (“exceeds”) - The Primary Model is “known” or “true” and the Reference Model is “unknown” or “false.” Represents the circumstance where the Primary Model is “doing better than” the Reference Model.
- **Fails to Meet Expectations** (“fails”) - The Primary Model is “unknown” or “false” and the Reference Model is “known” or “true.” Represents the circumstance where the Primary model is “not doing as well as” the Reference Model.

### 3.4.3 Filters

A Filter takes the full learner model and selects those elements most likely to be of interest to the student. Most notably, a filter which hides any material not pertinent to the “current” course is required.

Filters map a baseline learner model onto an alternative topology (Topology Mapping goal). At the simplest level, they select a subset of learner outcomes while retaining a fundamentally similar structure. However, they can also be used to select a set of learner objectives and map them to an entirely different topology.

Possible variant filters might show:

- The entire topology of the learner model vs. just that part represented by the current SPOC or MOOC.
- The course topology organised either by topic or by the week in which each learning objective was initially taught.
- That part of the course that had been taught up until a particular date.
- Just that part of the course that is assessed by a particular assessment instrument.
- Only just the easiest or hardest parts of the course.
In practice filters are used almost entirely for limiting the student view to the current course; reference models are also suitable for some of these applications.

As long as individual learner outcomes map cleanly (that is, all mapped entries are actually present in the OLM on the server), this permits two entirely different views of the same baseline learner model. In turn, this permits two courses teaching similar (overlapping but not matching) material to reference the same learner model.

Figures 3.2 and 3.3 illustrate the effect of applying a filter. Figure 3.2 shows the complete topology of our learner model, including all 1276 learner objectives defined. Figure 3.3 limits the viewed portion to just that portion of the ontology which is covered in the MOOC.

Use of the Reference Models and Filters is illustrated with additional examples in later chapters of this thesis. Discussion is deferred to prevent confusion due to as-yet-unexplained elements of the visualisation. Implementation is discussed in Chapter 5.

Use of Filters simplifies shared use of the learner model by multiple applications or MOOCs, as the student view of the learner model may be limited to just that portion of the model relevant to the course.

In practice, the set of Learner Objectives defined in our chosen filters overlapped very closely with the sets used for our template Reference Models. Since we were only working with a single SPOC, the topology mapping capabilities of the system were largely redundant, although a couple of these were implemented to demonstrate the capability.

As an example of how a Filter may be used to present an alternative topology,
Figure 3.3: OLM view with MOOC Evidence Filter applied

Figure 3.4 illustrates what our “default” (or full) ontology looks like when its ACM and non-ACM components are separated.

While the scale conceals this to some degree, the restructured OLM has broken out the supplementary learning objectives (left, under “Additional”) from those learning objectives present in the original ACM curriculum.

Use of Personis’ “Resolvers” (discussed further in Chapters 4 & 5) permits evidence to be filtered by time. This allows the user to view their OLM as it was at a chosen point in time, and to compare against a Reference Model as it existed at a chosen point in time.

### 3.5 Review of Aims

Our design to this point satisfies several of our design goals, as outlined in Table 3.1.

We expand on the use of these components in Chapters 4 & 5.
Figure 3.4: Example of Mapping to Restructure Model Topology

<table>
<thead>
<tr>
<th>Goal #</th>
<th>Description</th>
<th>Design Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Overview</td>
<td>OLM Visualisation + Filter for Learning Objectives concerning target Learning Activities</td>
</tr>
<tr>
<td>2</td>
<td>On-track</td>
<td>Compare student OLM against Curriculum OLM with time filter for current week</td>
</tr>
<tr>
<td>3</td>
<td>Behind</td>
<td>As for #2</td>
</tr>
<tr>
<td>4</td>
<td>Activity-Type-Progress</td>
<td>Apply a Filter for selected activity type</td>
</tr>
<tr>
<td>5</td>
<td>Act</td>
<td>Discussed in Chapter 4; based on comparison of student evidence store against “Ideal” model</td>
</tr>
<tr>
<td>6</td>
<td>Openness</td>
<td>Use of Personis server platform</td>
</tr>
<tr>
<td>7</td>
<td>Selectivity</td>
<td>Filters</td>
</tr>
<tr>
<td>8</td>
<td>Topology Mapping</td>
<td>Filters</td>
</tr>
<tr>
<td>9</td>
<td>Comparison</td>
<td>Reference Models</td>
</tr>
</tbody>
</table>

Table 3.1: Portions of Conceptual Model for each Requirement
Chapter 4

Design of the OLM

The first step in creating the OLM required by our thesis statement is to create an OLM optimised for use with MOOCs in a lifelong learning context.

Design of our OLM comprises two parts. The first part is constructing the underlying ontology for the learner model to be used. The second part concerns the visualisation presented to the student.

An earlier interface, *qv*, was later abandoned. *Nqv* is discussed in an appendix.

4.1 Representation of the Learner Model

Personis was chosen as our Learner model server platform. Source was readily available; data may be stored and retrieved easily; and it had been used previously in other OLM implementations [Apted et al., 2004, Uther and Kay, 2003].

Finally, Personis’ use of Resolvers permits variant interpretations of available evidence. The range of resolvers (and criteria used in their design) is described in Chapter 5 of this thesis.

Resolvers are Personis’ mechanism for permitting the evidence stored in a learner model to be interpreted differently depending on the needs of the application (or the student). Personis’ own data store includes only evidence for each learning objective; it does not store final, “resolved” values.

Opening these variant interpretations to the student was seen as important as it permits the student to view their learner model with multiple different priorities in mind.

As an example, linear video evidence has in the past been found to have low effectiveness in achieving learning outcomes [Zhang et al., 2006], a result supported by our own analyses.

However, a student may wish to confirm whether they have watched the video which reviews a particular topic to avoid over-review of known material. If the topic of a video is not also assessed by the MOOC, a resolver which
correctly weights video evidence as ineffective will not reflect that material has been seen. If video evidence is weighted as altogether ineffective, the OLM will never show that learner objective as having been met.

By using a Resolver which does not discard video evidence, the student may see those learner outcomes for which they have reviewed the material - and those for which they have not.

The standard “Resolvers” included in the open source Personis package do not provide a facility for reviewing evidence within a particular time window. This feature was desirable to permit the student or educator to review progress as at a particular time. The new Resolvers were written to include this capability.

4.2 Design of the Full Learner Model Ontology

The first step is to design the curriculum for our course. This is best done using a tool as as Gluga’s ProGoSS [Gluga et al., 2012] which allows all learning objectives to be outlined and described, as well as permitting assessment instruments to be described in terms of the learning objectives tested.

Ideally, we start with a suitable reference curriculum such as the ACM model undergraduate curriculum [ACM Joint Task Force, 2013] or version 7.5 of the secondary Australian Curriculum [ACARA and Authority, 2015]. I will refer to this as our “model curriculum”.

All learning objectives from our proposed course that are addressed in the model curriculum are mapped against the existing Learning Objectives. However, it is quite possible that we will wish to include topics outside of the model curriculum. For example, the ACM Curriculum defines one learning objective as follows:

Field : Computer Science

Subject : Operating Systems

Topic : Overview of Operating Systems

Tier : 1

Learning objective : 5

Description : Identify potential threats to operating systems and the security features design to guard against them.

A course teaching UNIX will normally have a section on use of file permissions. This topic overlaps with the ACM learning objective, but is still distinct, typically requiring additional information on use of the chmod, chown and ls commands from the UNIX command line.

Full coverage of the ACM item in the context of UNIX would also require information about the su and sudo commands as well as basic information about packet filtering using kernel features such as iptables.
However, coverage of the same learning objective in a Windows context would require coverage of Windows file permissions, the Windows Firewall, Microsoft’s reference documents for securing the platform, and several other topics.

We see that the ACM objective is fairly broad. It could easily apply to Windows or even Android security. However, what our SPOC teaches is primarily use of the `chmod` command. Learning use of this command does not imply a deep knowledge of security; similarly, it is possible to have extensive knowledge of data security but know nothing of the `chmod` command. The ACM learning objective provides insufficient atomicity for a course designed to teach material specific to UNIX or Windows. As such, *it is necessary in this instance to supplement our baseline curriculum with additional learning objectives specific to a particular MOOC.*

There are numerous other cases where this is necessary for this particular SPOC. The degree to which a curriculum will overlap with a course is highly dependent on the course and the curriculum; in the case of a mathematics curriculum, for example, we would anticipate a much greater degree of overlap.

We may alternatively find that the overlap is quite poor. As the ACM curriculum is, perhaps deliberately, highly environment-agnostic, it cannot be used for learning objectives which are, in fact, specific to a particular environment.

In order to facilitate sharing of the OLM between courses with overlapping material we use a fine-grained learner model. A course topic concerning UNIX file permissions will usually teach reading of file permissions with `ls` along with changing them with `chmod`, but may defer material on changing ownership for a later topic that covers `chown` and `chgrp` together.

Using such a fine-grained learner model introduces problems. Narrowing down our learning objectives reduces the number of learning objects for which observations may be used to populate each part of our learner model. Nevertheless, this approach was chosen because, while it is possible to summarise a group of learning objectives under a common category, thus incorporating all available evidence, it is not possible to take a single summary measure and infer the state of its components.

A more finely grained model also permits different choices in summary measures. A learning objective for “Knows the chmod command” can contribute to summary measures for “Knows Basic UNIX Commands” and for “Understands the permission model for security” where the two larger categories cannot be used to infer any information about each other.

While the current Filtering approach only maps single learning objectives, introducing this form of summary or aggregate representation is not expected to be unduly difficult.

Increasing the number of individual learner objectives also allows greater precision when referring the student to particular resource material. If desired, we can use the viewer’s filtering capabilities to hide those parts of the learner model that are not of immediate interest.

Exactly how this breakdown is handled is a matter of judgement, but if an existing learning objective exists in the current expanded form of the learner model, re-use can avoid duplication and facilitate cross-comparison.
We place new learning objectives within the overall curriculum to take advantage of existing structures in the model curriculum’s ontology. This simplifies finding a learning objective about a particular topic.

One additional issue raised by this process is how “truth levels” are to be mapped between different representations. How do we map between the ACM taxonomy (Familiarity, Assessment, Usage) and the Bloom taxonomy (Knowledge, Comprehension, Application, Analysis, Synthesis, Evaluation)? Any mapping chosen will be inexact; the ACM’s “Usage” arguably covers all four top levels of the Bloom taxonomy. We cannot answer this definitively, but suggest that the chosen model curriculum be used as a baseline; for example, for Computer Science, use of the ACM taxonomy should be used. Choice of this characteristic is dictated by the available data and the purpose for which it will be used; in some cases a cross-mapping between ontologies may not be practicable at all. In such cases the OLM must, of necessity, include separate learning objectives for each scale used.

A MOOC designed from a given curriculum will already have its Learning Objectives defined for each Learning Object. Where no such formal mapping is available, it is necessary to examine each Learning Object or Assessment Instrument and determine the corresponding Learning Objective. If the Learning Objective is not represented in the provided curriculum but must be tracked, the curriculum must be supplemented in the manner outlined above.

This process results in a core learner model based on our Model Curriculum, plus a number of supplemental learning objectives for each course. Once the curriculum has been converted to a learner model, use of the Reference Model and Filter supplemental features will facilitate examination of the baseline and supplemental learning objectives present in our curriculum.

The Learning Objectives so chosen are then used to create our Learner Model. Supplementary Learning Objectives are added, where possible, to corresponding, broader topics under our model curriculum.

4.3 Design of the Visualisation

An OLM has two core components: the data store which tracks the students’ knowledge and characteristics, and an interface or visualisation which actually opens the learner model for learner interaction. We discuss here the design of this visualisation.

We seek a design which facilitates unification of course design principles with a scalable lifelong OLM framework. As lifelong learner models may be quite large, we would like it to be scalable. To facilitate comparison between students and evaluation against a fixed curriculum, it should permit comparison of learner models. To make it clearly useful as a tool for the student, the student should be able to use it as a tool to guide further learning.

Per Table 2.1 in our Background chapter, most currently published visualisations fall short in at least one of our three criteria.
• **Scale**: Learner models beyond a certain size become unwieldy. In particular, most visualisations are fundamentally *linear* in presentation and so the size of the model that can be presented effectively is limited.

• **Comparing Models**: Only Mastery Grids permitted comparison of progress between students.

• **Focus on Learning**: Most of the visualisations not specifically targeted at learning had no obvious way to include references to learning resources.

To address these issues, we attempted to find a practicable alternative visualisation.

### 4.4 Reference Models and the Learner Model

Our overall approach is as shown in Figure 4.1.

![Figure 4.1: Integration of components for the OLM](image)

This diagram corresponds to Box C in Figure 3.1 in Chapter 3. The OLM tracks a “Primary” learner model and an optional “Reference Model”. The Primary Model is conventionally that of an individual student. The Reference Model may be a “**User Reference Model**” based on a student or a notional student (such as expected progress at a given point in the course). It might also be a “**Curriculum Reference Model**”: an assessment instrument, or a selected portion of the curriculum. The Primary Model may also be populated with one of these alternative categories.
We take advantage of Personis' "Resolver" feature to allow the student to allow the evidence stored in the OLM to be interpreted in any of several ways. By selecting a suitable Resolver, the student may choose to examine the state of their learner model at a given point in time, or to favour or discard certain evidence.

The system combines the resolved values from the two Learner Models into a common, comparison model which stores both. This is done to facilitate comparison of the two learner models by the visualisation engine. The comparisons resolve each component to one of the four possible integrated values ("true", "false", "exceeds" or "fails") for display.

The resulting learner model may still be arbitrarily large, containing the whole of the original ontology. As a final step, we apply a filter (or map) which selects a small subset of the full learner model for display. The filter may also be used to remap the topology of the learner model as displayed.

This mapped learner model is then passed to a visualisation engine. The visualisation engine permits any of several different representations via a drop-down selector. We chose the "circular treemap" as default intuitive representation of the learner model structure while retaining good scaling.

This design permits the default visualisation to be supplemented by different visualisations which may offer additional advantages, while still taking advantage of Reference Models and Filters if desired. In so doing we allow the learner to take advantage of the strengths (and offset the weaknesses) of a selected visualisation.

Evidence items are stored in the remapped ontology to permit detailed examination by the student.

The system also stores an "Ideal Student" learner model with exactly one instance of every possible evidence item represented by a learning object. This model is then used as a reference for all available learning resources. When a student asks for additional learning resources for a learning objective, their evidence list is checked for matches against the list in the "Ideal" model and they are informed of any learning resources that they may have missed.

Construction of the "Ideal" model is straightforward as we already have a mapping between learner objectives and learner resources as part of our work in mapping from MOOC learning resources to learner objectives.

Thus we use the ontological structure of the Learner Model for three purposes. Firstly, it stores our primary Learner Model of interest. Secondly, it is used to construct multiple Reference Models for comparison. Finally, by way of the Ideal Student model, the structure serves as a reference for available learning resources.

4.5 MOOClm

The core interface consists primarily of a title bar at top, a control panel on the left, a viewing area using the bulk of available space, and a message/status bar at bottom. Figure 4.2 shows the overall interface layout.
The title bar includes a simple “help” button which gives an overview of navigating the interface as well as a “search” function. Text entered into the Search bar will result in all learner objectives with a description matching the supplied text being highlighted with a modified border and with the learning objective’s label being shown in blue. Figure 4.3 shows the result of searching for the string “for loop”.

The Message Bar shows any system messages (which are also shown via popup dialogues). When zoomed in, it normally also shows the path (relative to the model root) for the current zoom context.

Elements of the Control Panel control details of what is shown in the primary viewing space.

4.5.1 OLM Visualisation

After some experimentation with alternate visualisations, we chose the “pack” visualisation provided by Bostock’s D3 Javascript library [Bostock et al., 2011] which uses a “circular treemap” representation. This visualisation uses nested sets and subsets of circles to represent hierarchical structure. As such, the (vertical and horizontal) space used in representing a set of N nodes is $O(\sqrt{N})$ rather than $O(N)$ vertical or horizontal space.

As a result, learner models in sizes on the order of thousands of items can be viewed without unduly compromising the visibility of the structure of the learner model.

As mentioned in our Background review, this representation does waste a significant amount of white space compared to many other hierarchical two-dimensional representations such Schneider’s rectangular TreeMap or the Vor-
Figure 4.3: Searching the Model
bonoi Treemap. However, this “wasted” white space provides clear visual boundaries between learning objectives and components, and therefore contributes to an at-a-glance understanding of the structure of the learner model and its parts.

To make successive depth levels of the visualisation clearer, the nested circles are shown in progressively darker shades of grey.

Figure 4.4 shows the view for our extended ACM model (the ACM curriculum ontology plus additions for the COMP2129 MOOC course).

The visualisation is not without its shortcomings. Text labels located across the centre of each “bubble” tend to overlap, and tend to obscure any underlying structure. This was addressed by a number of refinements.

- Only the next-from-top level is labelled, so only the largest parts of the learner model are immediately visible.
- The labels are placed at the top of each circle, rather than in the middle, to reduce overlapping.
- The labels may be slight slanted via the Visualisation selector, again to reduce overlap.
- Tooltips are used to allow the user to examine any other item.
- Clicking in any “bubble” for part of the model zooms into that part.
- The use of “Filters” described earlier permits the view to be limited to a selected subset of learning objectives, eliminating some of the issues with the scale of the visualisation.
4.6 MOOClm: Mapping and the Reference Model

In addition, we wished to provide additional facilities to facilitate alternative uses of the learner model. We wanted to permit the user to:

- **(Topology Mapping)** Examine alternative topological views of the same learner model, so that the structure of different courses teaching the same learner outcomes could use a common learner model.

- **(Selectivity)** Restrict the view of the current course to a list of topics of interest. This is essentially a special case of our first requirement.

- **(Comparison)** Compare the learner model with that of another student.

- **(Comparison)** Compare the learner model with a specified target or Reference Model.

The final implementation enables these activities by permitting selection of the Reference Model and one of several Filters, as outlined earlier, in addition to the “primary” learner model.

“Filters” are to be implemented via a map defined for each desired derivative of the model’s topology. Filters are defined in terms, first, of structural elements (“branches” of the hierarchical tree) and, second, of placement for each learning outcome of interest.

This can be logically extended to also map taxonomies, so that a learner model created using the ACM Taxonomy may be viewed or changed using the Bloom taxonomy, for example. Another likely extension is the definition of aggregate entries incorporating multiple learning objectives.

Basic uses of the mapping facility are to limit the view down to just those learner outcomes taught in the current course (or another desired subset), and to map between different course structures.

Figure 4.5 shows an example of our design. This shows a view for our learner model with a filter applied to limit the section in view to those learner outcomes tested within the COMP2129 SPOC.

Our second addition is that of a “Reference Model”, a second learner model with the same basic structure for comparison purposes. As discussed in Chapter 3, this can be an actual learner model (a User Reference Model of another student or of a notional “ideal” student; or it can be a Curriculum Reference Model.

This permits a student’s learner model to be actively compared against other students or against selected curriculum or assessment targets.

Furthermore, we can apply this extended use of the Reference Model back to the primary model, permitting use of the same tool in comparing curricula. Our visualisation then becomes a tool not just for viewing a learner model, but for evaluating performance of both the student and the curriculum and its assessment instruments.
Figure 4.6 shows the view for our learner model, with a filter applied to limit the section in view to those learner outcomes tested within the COMP2129 SPOC.

Introduction of the reference model does require some additional flexibility in presenting the learner model. Whereas showing a single learner model requires at minimum two colours (known/unknown), adding the reference model adds an additional dimension (the truth/falsity of the reference model).

While one possibility here was to combine colour with alternative graphic elements such as border thickness or pattern, this would have made it more difficult to immediately locate particular patterns of immediate interest.

By way of example, it is easier to locate a blue circle amongst red, black and white circles than it is to locate a white circle with a thick outline in a display which includes many white circles with a thin outline. As such, only a single dimension (colour) was used to show possible values, with a colour picker included to assist the user in emphasising particular elements.

As such, representing the reference model requires two additional colours, nominally labelled as “Exceeds Expectations” (“exceeds”) and “Fails to meet expectations” (“fails”).

In a broader context, these indicate, for “exceeds”, that the learner model marks an outcome as “Known” where the Reference Model has it “Unknown”; or for “fails”, the converse.

While this representation offers considerable power, alternative use of colour scales, such as indicating certainty levels or taxonomy knowledge levels, is thereby lost. However, as these are rarely explicitly represented in any case, we judge that the loss is bearable.
This double-loading of colours risks confusion for users without a thorough understanding of how the colours and truth values should be interpreted. This problem is difficult to avoid; it can be mitigated to a limited extent by using overlapping colour schemes (for example, setting only the “exceeds” option to black and all others to white).

We judged that as the additional complexity was only present when a reference model was used, it could be ignored for simple uses of the interface, while expert users would understand the distinctions being made.

One other possible approach is to display two models side by side, but this could be reproduced by simply running two browsers side by side, and lacks precision in comparing specific elements. A third option is to blink between two models, which in addition to being visually distracting does not distinguish well between the Primary and Reference models.

4.7 Resolver Tuning

The “Filter” and “Reference Model” controls modifying (respectively) the ontology of the model view and, optionally, the learner model against it is being compared.

Use of the Personis model server created an additional opportunity. Personis relies on use of “Resolvers” to determine the final value of learner model components. This capability allows evidence to be weighed differently depending on the needs of the viewer. For example, a resolver might choose to ignore video evidence, or require a higher or lower standard of evidence, or to ignore evidence before or after a given date.

While Resolvers are to a large degree an implementation issue, the additional
capabilities they offer extend the available options in presenting a learner model and so are significant from a design perspective.

With these capabilities in mind, we propose three additional selections for the interface.

The first allows the user to select one of several resolvers to fine-tune the standard of evidence used.

The resolvers may choose between extremely trusting (“any evidence is sufficient”) or extremely pessimistic (“only if the student got all answers in short-answer questions right at least once”).

The second and third resolver tuning options provide an “As of Date:” option for the primary and reference models. This permits the viewer to omit evidence items after a given date, and resolve the model accordingly. While Personis does natively provide an option for date filtering, this option is only used when filtering the evidence passed through to the client; it does not modify the resolved values of components in the model.

Adding the option for date filtering when resolving the model allows the user to view and compare the progress of students over time. Having the same option for the Reference Model then allows the viewer to compare the progress of the student with that of a predetermined set of progress checkpoints.

For example, we can look at a student’s progress at the end of March 2014 and compare against where they should be according to a reference curriculum. Alternatively, we can compare a reference model against itself to compare how much of a course would have been taught at a given point in time, or to show the learning objectives that were taught during a particular interval.

For this facility to be useful, the dates in evidence present in the primary and reference models must be recorded accurately. We corrected a small bug in Personis which initially prevented this.

4.8 Simple vs. Expert Interfaces

While these additional tools extend the potential power of the toolset considerably, the resulting complexity also adds considerably to the potential for confusion amongst novice users.

To mitigate this, we add a simplified interface, in contrast to the “expert” interface in which all options are present.

Figure 4.7 shows a side-by-side comparison of the Simple and Expert interfaces.

The “Simple” interface lists a handful of basic comparison options, plus login name and password and colour controls. The OLM viewer defaults to this interface when initially started.

Selection of options within the “Simple” interface modifies the same parameters used in the “Expert” interface so the two can be switched cleanly for fine-tuning, if necessary.
4.9 Component Selection and Display

Previously we have described tuning options for the overall view of the learner model. Here we outline the visualisation’s options for viewing of individual model components.

One of the weaknesses of the circular treemap visualisation is that text labels on small parts of the model will tend to overlap or obscure each other. In addition to only showing label text on relatively high-level portions of the learner model, a tooltip or hovertext was added for each component and leaf, giving the name of that particular node and the text description included for the node within Personis.

In addition, clicking on any high-level (non-leaf) component will “zoom in” to that part of the model. This allows parts of the model to be viewed in greater detail. Figure 4.8 shows the result of successively “zooming” through the SoftwareFundamentals and DevelopmentMethods parts of our MOOC curriculum.

Clicking on a “leaf node” - an actual learning outcome - brings up a detailed display of information concerning that outcome, as well as some options to view learning resources and to request that the value of that learning outcome be modified. Figure 4.9 shows a sample display, here for the notional C “Strings” data type. Figure 4.10 shows the same display, scrolled down to where additional evidence items derived from MOOC exercise sets are listed.
Figure 4.8: Zooming into the Learner Model: When viewing SoftwareDevelopmentFundamentals, clicking the DevelopmentMethods bubble zooms into DevelopmentMethods.
The first item in the display is the base name of the learning objective. The tooltip for this is the long path for the learning objective; in this case, “ProgrammingLanguages,C,Types,Strings”. This is followed by the description of the learning outcome as found in the Personis OLM store.

The next item given is the “value” of the learner outcome using the taxonomy for the learner model. The tooltip for the value is a long description of the value, as found in the original taxonomy. For example, the tooltip for “Familiarity” is “The student understands what a concept is or what it means. . . .” and so on, as outlined in the ACM curriculum document.

The next item listed is the name of the resolver used within Personis to generate the truth values. While in this instance the default resolver was used, the description for the resolver may be of arbitrary complexity. For example, the description for “edxweak” is simply “Any evidence is sufficient”, but the description for “edxcareful” (which fits assessment results against a normal curve) is “Correctly answered sufficient relevant questions that we are 90% sure you know the topic.”

The resolver is described at this point because a student using the “Simple” interface may not be aware that the evidence may be handled in different ways. In particular, they may not understand why certain evidence, in particular video evidence, is ignored.

After the resolver is given, the system gives a link to a second dialogue box which displays possible learning resources. This is outlined in the section on Learning Resources.

The final item included is a list of evidence items (“Learning Data”) found in the Personis database. If a time filter was applied, it is also used to filter this list. By default, the display includes a timestamp, a description of the evidence, and the knowledge level stored for the evidence item.

Hovering the mouse over an evidence item displays a (not particularly intuitive) formal description of the source of the listed evidence item, as stored in the original Personis learner model. For example, for the top item listed, this is UniSyd:COMP2129_MOOC:Week 2:C Aggregates and Pointers - Pointers[415-634]

The numeric range [415-634] is an offset to the relevant section of the video, measured in seconds.

Finally, at the bottom of the dialogue box are options to modify the learner model. “I KNOW This” introduces an “explicit” evidence item (in Personis nomenclature, indicating explicit user feedback) indicating that the learning objective is known. “I DO NOT Know This” similarly adds an evidence item with an explicit “Unknown” value.

“Close without action” closes the display without making any changes, as does simply closing the window.

Figure 4.11 illustrates what the evidence looks like with a different resolver specified where “I KNOW This” has been selected.
4.10 Learning Resources

Clicking on the link for “Click here to pop up a list of available learning resources for this item” will bring up a display similar to that in Figure 4.12.

The list breaks down evidence into those items which have been visited by the owner of the learner model previously and those which have not yet been visited.

Video resources include both the video reference on YouTube (extracted from the edX Mongo database) and a link to the page where the described video is stored. YouTube links will take the user to the correct video segment; unfortunately this is not an option for external links to edX.

Links for problem sets also take the user to the relevant page of the MOOC rather than the individual problem, due to limitations in how edX handles links from external sources.
Providing this list of learning resources is intended to be of use to both the student (who can find resources they have not used) and also to the educator (who can review all learning resources specific to a learning outcome, and if necessary revise them accordingly.)

4.11 Directive to New Learning

The interface also includes one minor element. It uses an internally stored list of learner objectives in the order in which they are normally learnt. The first such learning element which does not resolve as known by the user is highlighted with red text and a modified border.

In most of these examples you will find this highlighted element is “DevelopedBy.”

This is intended to direct the student to the next area of likely interest.
Chapter 5

Implementation

We have described our design for an OLM for use with MOOCs in a lifelong learning context. In this chapter we describe the implementation of the OLM, go through some examples of using the OLM to answer our core questions, and also the steps required to integrate our MOOC with our OLM.

We begin with an overview of our system architecture, then continue by describing the process used to populate the OLM for which our ontology was chosen in Chapter 4. We then describe implementation details of the OLM visualisation including basic details on how the frontend and interface to the Personis backend was handled.

We also describe how the OLM directs the user towards suitable learner resources in the MOOC and also how the OLM itself may be launched from the MOOC.

Finally, we discuss our statistical analysis of the student data set.

5.1 System Architecture

You may wish to refer here to the design outlined in Figure 3.1 in Chapter 3.

The process of populating the Learner Model follows the procedure indicated in Figure 5.1.

The actual MOOC software (OpenEdX) runs on a dedicated platform. EdX has a Learning Management System (LMS) for use by students and a Course Management System (CMS) which is used to assemble, modify and evaluate courses. The course structure is stored in a Mongo database; logs are generated as text files.

We use the course structure (including information about objects in the MOOC) in a set of scripts, primarily written in Python, to parse the server’s activity logs against a stored database linking each learning object with a corresponding list of learning objectives.

These correspondences are then used to inject evidence into an Open Learner Model for each user, stored ona Personis server. The system also stores several
Reference Models for comparison, and has multiple Resolvers to allow alternative interpretation of the collected evidence.

This Learner Model is then made available to the learner for interaction and correction.

## 5.2 Constructing and Populating the OLM

Construction of the ontology for our OLM was described in Chapter 4. This process generates, as a side effect, a list of Learning Objectives associated with the course’s Learning Objects or Assessment Instruments.

We have chosen the ACM 2013 curriculum as our model. This curriculum was chosen as it was published by perhaps the best-recognised professional I.T. body; is quite recent, having been finalised only shortly before this thesis commences; and explicitly includes learning objectives for all defined topics.

Working with our COMP2129 SPOC, we supplemented the original 1105 ACM Learning Objectives with 170 additional objectives specific to material covered in the MOOC but absent or under-represented in the ACM curriculum. We do not include the Full ontology in this thesis; that part of it which is specific to our MOOC is included as Appendix D.

We must then parse the server’s logs to generate a log of evidence, then insert that evidence into our OLM.
5.2.1 Parsing Evidence Logs

Internally, the OpenedX server logs all events in a “tracking log”. This log includes all attempts at self-assessment questions as well as extensive, but not exhaustive, coverage of video viewing activity, as well as other activity within the MOOC.

As the tracking log uses an internal edX identifier for each logged entry, the association of a particular self-assessment exercise or video with the logical structure of the MOOC as seen by the student and educator is not immediately apparent.

The first stage, then, is to map the internal identifiers used by edX in its Mongo courseware database to something more intuitive. Fortunately, it proves to be relatively straightforward to do this, although the process is complicated by the fact that the “JSON export” for Mongo does not generate compliant JSON output. For example, the JSON may include a property such as:

```
"edit\_info" : {
    "edited\_by" : -1,
    "subtree\_edited\_on" : ISODate("2015-02-27T03:15:52.085Z"),
    "edited\_on" : ISODate("2015-02-27T03:15:52.085Z"),
    "subtree\_edited\_by" : -1
},
```

This requires some minor on-the-fly fixes to ensure that the file is compliant for use in standard libraries.

The Mongo database includes "children" entries for identified objects. This information can be used to create a logical path for each object in the database. Entries in this mapping will look something like this:

```
i4x://SIT/COMP2129/vertical/2549935d9f9b4aaeb6fc96725c6e441:
    Operating Systems and Machine Principles/Week 2/
    C Aggregates and Pointers/Scope quiz
```
```
i4x-SIT-COMP2129-problem-0696d3f1a995a88e575fe98dfd7c:
    Operating Systems and Machine Principles/Week 6/Shell Quiz/
    File Permissions/Multiple Choice #8
```

The format on the left varies as several variations are used within the edX tracker logs. To save later processing, mappings for all used formats are generated.

Our mapping of learning objects to learning objectives can then use the verbose form, such as Operating Systems and Machine Principles/Week 6/Shell Quiz/File Permissions/Multiple Choice #8 along with identification of corresponding learning objectives.

In the case of video segments, we also include timestamp information for the relevant segment of the video. This allows us to track learning objectives to video segments. However, in practice this is probably unnecessary for two reasons. The first is that edX does not track video views with great accuracy;
except for active intervention, such as manual seek events, edX typically (but not always) checkpoints video views at two minute intervals. The second reason is that our statistical analysis demonstrated no significant effect from video views on meeting learning objectives, so the value of such data in parsing truth values from evidence is minimal.

A sample entry for a video segment would then look like this:

```
[ "C Basics/Control/Control", "50,535",
  "Education/ComputerScience/SoftwareDevelopmentFundamentals/
   FundamentalProgrammingConcepts/Tier1/01",
  "Education/ComputerScience/SoftwareDevelopmentFundamentals/
   FundamentalProgrammingConcepts/Tier1/07" ],
```

where the first field is the video's logical path within the course and the second is the part of the video which covers the learning objectives defined thereafter.

The same file for an exercise simply omits the video segment:

```
[ "Unix Basics/Files/Multiple Choice #7",
  "Education/ComputerScience/OperatingSystems/UNIX/BourneShell/
   Core/FilePermissions"
],
```

However, manually editing such a list is quite tedious and error-prone. The problem is fundamentally one of creating an N-to-N database mapping; as such, a simple database tool supplied with the correct object sets for learning objectives would be preferable, or a simple extension to edX proper or to ProGoSS.

We then parse the edX event logs for entries indicating activity for each learning object.

### 5.2.2 Parsing Evidence

The above process gives us a list of learning objects, including internal edX identifiers, and corresponding learning objectives.

The next step is to parse the edX logs. While these logs are nominally in standard JSON format, actual choice of fields in log entries meant this process was nontrivial. In many cases, the bulk of a log line consists of a duplicate of the HTML sent to the user. In others, simple information such as whether a question is answered correctly is missing.

The standard edX tracking logs are somewhat duplicative while omitting other useful information entirely; parsing them was in itself nontrivial. While these issues were later addressed by open source code described in the literature (such as by Pardos & Kao [Pardos and Kao, 2015]), at the time of original development rationalising the edX logs presented a serious problem.

### 5.2.3 Construction of Reference Models

As we already had a database of learning objectives and corresponding learning objects, constructing our reference models was relatively straightforward.

The “typical” student OLM was generated simply by choosing a representative student, obfuscating all data that could be identified against the student,
and generating JSON data corresponding to the format used internally by the
OLM visualisation.

Additional Reference Models were constructed by selecting the relevant sub-
sets of learning objectives and populating target Reference Models accordingly.
The reference model was initialised as “empty,” with all learning objectives de-
fined but no evidence attached, from the same definition file used for students.
Evidence was injected directly by selecting the evidence for the set of learning
objectives or learning objects relevant to each reference model. A short script
calling the relevant python libraries for Personis would then inject these into
the model.

5.2.4 Construction of Filters

Filters map a baseline learner model onto an alternative topology. At the sim-
plest level, they select a subset of learner outcomes while retaining a fundamen-
tally similar structure. However, they can also be used to select a set of learner
objectives and map them to an entirely different topology.

Most of these maps are pre-defined in an attached Javascript inclusion for
speed of loading, but any map which is referenced but not already present is
loaded dynamically using AJAX. The files are formatted as JSON arrays.

The raw text of the filter, as implemented, consists of “add” entries and “map”
entries, structured in JSON format. “Add” entries create structure, defining the
“branches” of the hierarchical tree to which the nodal learning objectives are
attached. An Add entry might look like this:

```json
{  "action": "add",  "tag": "OperatingSystems/Concurrency/Tier2",  "desc": "Tier 2 Learning Outcomes in OS/Concurrency"}
```

A “map” entry maps between the location of a learning outcome in the base-
line ontology stored within Personis and an alternate location under structural
elements already defined by “add”. A “map” entry might look like this:

```json
{  "action": "map",  "src": "Education/ComputerScience/OperatingSystems/Concurrency/Tier2/05",  "dst": "OperatingSystems/Concurrency/Tier2/Synchronization"}
```

This allows learner outcomes to be logically shifted around the ontological
topology viewed by the user. Since those items in the original ontology without
Corresponding “map” entries are omitted from the view, it can also be used to
filter out parts of the OLM which are not of interest. This can be important
when dealing with a course which is dealing with a small part of a large syllabus
such as the ACM curriculum.
As most filters merely present a subset of the larger ontology, constructing these was straightforward. The desired learning objectives were selected. Any “parent” entries above these (such as the OperatingSystems branch) were recreated using “add” entries, then learning objectives were appended using “map” entries.

5.3 Implementing the Visualisation

In order to eliminate the need for custom software on the user’s desktop, we chose to implement the visualisation using HTML and Javascript, with interfacing to the Personis database via a small number of custom CGI scripts coded in Python.

Logically, the visualisation is implemented in four parts. These are the core interface, the visualisations, the OLM backend, and a Javascript object holding a representation of the primary and reference learner models and the current mapping.

5.3.1 Core Interface

While the “home page” of the interface is actually in PHP, this is done solely as a measure to permit pre-authenticated access from the edX MOOC. edX includes support for external learning resources, authenticated using the LTI (Learning Tools Interoperability) protocol to permit students to be referred to external learning resources. The PHP code on the home page checks the LTI data passed through to it in order to permit pre-authenticated access to the password-protected Personis database.

Once this authentication is complete, PHP is not otherwise used.

Layout is performed using standard HTML formatting elements and places using CSS3. Interactivity is handled using JQuery event handlers, using AJAX where necessary to load external data asynchronously.

Learner models are loaded via functions described in the OLM Backend.

5.3.2 OLM Backend

The OLM Backend includes two primary components. Filters are simply loaded from included javascript files at runtime. If a filter/map is accessed which is not so defined, the map is retrieved using an AJAX query from a directory set aside for that purpose.

Reference Models are stored within Persona proper in order to facilitate use of specialised Resolvers and date tuning. This eliminates the need to replicate evidence filtering and evaluation from Persons’ Python-based Resolvers in a form directly usable by Javascript. The OLM CGI scripts allow read-only access to these OLMs; access is further restricted via a password used in the python backend CGI script but not found in the HTML loaded into the user’s browser.
Interfacing with Personis is handled via two CGI scripts. The first loads a learner model using provided user credentials along with supplementary information such as the resolver selection and date filtering information. The second CGI script is used to write user modifications to the OLM back to the Personis database.

Modifying the interface for alternative OLM hosting backends (i.e. other than Personis) is expected to be trivial, although of course most OLM backends do not use a parallel to the Personis Resolvers.

Unfortunately since the browser stores a complete snapshot of each learner model, the actual loading process can be quite slow. A future implementation may address this by applying the topology mapping/filtering function in the server-side CGI scripts.

5.3.3 Internal Learner Model Representation

The Internal Learner Model Representation includes Javascript dict structures which store arrays containing learner model data as loaded using the OLM Backend.

The object includes the following structures:

- **um_original** - the raw form of the primary learner model, as loaded from Personis.
- **um** - The primary learner model, restructured according to the loaded map file. Restructuring includes comparative information against the Reference Model.
- **ref** - The reference learner model, as loaded from Personis.
- **ideal_ref** - A copy of the “ideal” learner model. This is loaded by default so that location data for all available learning resources is present; it may be thought of as a learning resource location reference.
- **map** - The currently used map or filter, as a Javascript dict.
- **readonly** - Flag set to 1 when loading a reference model as a primary model.

It also defines the following methods:

- **applymap** - Apply a new map to the learner model.
- **cleanup** - Method which scans a specified internal learner model and sanitises it, setting values such as the number of sub-elements in each parent.
- **compare** - Loads truth values from the Reference Model into the mapped primary model. Returns an error if there is a topology mismatch between the two models.
• **viewtoggle** - Toggles a model element between visible and invisible; this is primarily in place for certain legacy visualisations.

• **redraw** - Calls the appropriate method for redrawing the current OLM visualisation. This references certain fields on the main page and changes the primary display between canvas and svg elements if needed.

• **master** - Loads the Primary learner model.

• **reference** - Loads the Reference model.

• **ideal** - Loads the Ideal learner model.

• **model_ro** - Sets the Primary model to be read-only, setting the readonly flag.

• **model_rw** - Sets the Primary model to be writable, resetting the readonly flag.

• **model_is_ro** - Returns the current readonly status of the model.

The file which defines the OLM structure also includes utility functions common to multiple visualisations. These include methods to encode the evidence for a component, to determine the appropriate colour for each node and to generate a verbose description of each “knowledge level.” The functions used to display evidence dialogues, generate learning resource lists and to modify learner objective values are also found in this module.

5.3.4 Visualisations

The interface permits selection of several visualisations, although at present most of these are disabled for user view to avoid confusion. Interfacing to these is handled via a method in the internal UM representation.

The “Bubble” visualisation primarily referenced in this thesis uses Bostock’s D3 library with its “pack” visualisation on an SVG object. Event handlers are included to handle the zoom/unzoom function and popup dialogues for evidence reports. Evidence reports are actually coded within the Internal UM Representation Model in order to facilitate re-use across multiple visualisations.

The visualisation as coded uses a slight gradation in higher-level fields so that deeper levels look darker; this was done to highlight the differences between each level, as a simple outline did not stand out particularly well.

There are also internal allowances to modify placement of the tags for each “bubble” and to display and modify the highlighted “next” Learning Outcome and any entries found using the Search functionality.

Unfortunately, at present reloading or modifying the model also loses any current zoom level.

The first of the alternative visualisations currently offered is a slight variation where the text for each bubble is tilted upwards at a fifteen degree angle. This attempts to reduce the degree of text label overlap. The second alternative is a
partially-implemented “Sunflower” visualisation, handicapped by a lack of text labels (which are shown in hovertext). The final alternative is perhaps the most usable, consisting of a “dendrite tree” displayed with the model root at the left. Figure 5.2 illustrates this view.

All visualisations were implemented using Bostock’s D3 library.

Figure 5.2: Sample of prototype dendrogram cluster visualisation

Addition of a colour picker permits the user to tune the colour scheme to emphasise or de-emphasise particular elements or to allow for different types of colour blindness. For example, if a student wants to explicitly see those areas where they are falling behind, they can pick black (or red) for “fails” (Fails to Meet Expectations) and white for all other colours.

The additional colours used for comparison of Reference models are hidden from view (via CSS properties) if no Reference model is chosen.

The Simple interface is a subset of the Expert interface, except in that some selections modify multiple parameters. As such, when Simple selections are made, hidden elements on the Expert panel are simultaneously adjusted (by modifying HTML form properties via jQuery) to match the new model parameters. This allows the user to make a selection via the Simple interface then tune it on the Expert panel.

Similarly, if changes are made on the Expert interface, the Simple interface is modified to the “Complex (Use Expert parameters)” option.
5.3.5 Privacy and State Tracking

The implementation used does not use cookies or server-side state tracking in any form, save for an initial handoff to the OLM from EdX using LTI. Instead, all state is stored either in the actual learner model (which must be re-authenticated at every access) or within the window state information tracked internally by the browser. This avoids any privacy concerns associated with use of cookies and allows a single browser to run any number of OLM windows in parallel.

Thus, if the user desires and has the appropriate authentication information for each learner model, they can switch between the views for any number of learner models without undue concern about accidental cross-contamination of data.

In principle this can be used to compare more than the initial pair of learner models, or perhaps using the time filtering capabilities of the OLM to view progress of student learning over time.

The drawback to this approach is that, since each window holds a complete copy of the learner model, the memory requirements can increase dramatically if multiple learner models are viewed.

5.4 Personis Backend and Resolvers

Interfacing to Persona was handled via CGI scripts present on the server. Currently the Persona data stores reside on the same server as the web server, but Persona’s network access interface allows the learner model stores to be kept elsewhere.

Initial work to use evidence on a weighted basis was based on accumulation of evidence to support a particular outcome. A calculated probability of the learning objective being unknown would start at 1; each additional piece of evidence was assigned a weighting according to its importance. When these factors were multiplied together, a result below a chosen threshold would mark the objective as “known”.

For example, assessment exercises might be given a weighting of 0.5 for correct answers and 1 for incorrect answers; video, 0.9 for the first view and 0.99 for subsequent views. These values were chosen so that two correct answers, or a huge preponderance of video evidence, would push the probability below the chosen threshold of 0.3.

This approach was abandoned when it was found that the results were both too generous and too lenient. Many learning outcomes were represented only in video evidence - in some cases, in quite short segments. As a result, most models generated were mostly “unknown.”

The next approach used was termed the “weak” resolver and used the principle of any evidence at all being sufficient. While this resulted in learner models that looked more interesting, as they contained many more “true” elements, the learner models did not accurately represent the state of learner knowledge. As such, learner models using this resolver are visually interesting but practically
useless.

The final approach chosen built upon our statistical analyses. We found that the average user mark was about 60%, but high-performing students did better, particularly on difficult topics, than poorer-performing students. Most importantly, harder topics would give better results for those students who did better overall, indicating that there was in fact some differentiation between learner outcomes.

This analysis was largely independent of the number of actual MOOC problems attempted by the student. That is, the percentage of correct answers for students who attempted many problems was about the same as that for students who attempted few problems.

Video evidence was ignored, as all statistical analyses showed no statistical significance at all of video views on final exam results.

As such, our final resolver looks at the percentage of correct answers in the MOOC questions specific to a learning objective. It then calculates a confidence interval using a mean of 60% and standard deviation of 14%; this is equivalent to a percentage threshold of 77.9%. Students who get a better result than this are judged as “knowing” that learning objective.

Our Resolvers included some additional logic, absent in the original Personis, to filter evidence before including it in the analysis. Personis does include some time filtering by default, but this is only applied to the returned evidence lists; excluded evidence was still being included during the process of resolving truth values.

The viewer includes encodings for ACM and Bloom taxonomies. As we only used ACM taxonomies, we could ignore clashes in taxonomy nomenclature (for example, the differing definitions of “Knowledge” between the Bloom and revised Bloom taxonomies) but if this work is carried forward we see a need to include a reference to the taxonomy used when displaying resolved learner objective values.

5.5 Integration with the MOOC

The viewer was designed and implemented on a private Amazon EC2 instance at [http://personis.ronnycoo.net/](http://personis.ronnycoo.net/) and partially integrated, after Ethics committee approval, with the MOOC instance on the University-run EC2 OpenEdX instance at [http://online.it.usyd.edu.au/](http://online.it.usyd.edu.au/).

The OLM evidence list includes links to pages where resources may be found on the MOOC as well as to the YouTube locations where the lecture videos are stored. The MOOC, following ethical approval, included a link to the MOOC, per Figure 5.3.

Calls to the OLM used the “Basic” LTI (Learning Tools Interoperability) standard to hand off authentication to the OLM via a PHP library checked on the OLM launch page. Following successful authentication, the PHP code on the server established a short-term private key to permit the authenticated user to access the system from the same browser.
Figure 5.3: MOOC launch page for the OLM (top and bottom)
Unfortunately while EdX does support LTI, its intended use is not for integration of tools of the complexity of MOOCm, so full integration was not deemed to be practical within the constraints of a Masters thesis.

Furthermore, the version of EdX in deployment had a bug which would often hide LTI interface button; these could be viewed using the well-known trick of highlighting invisible text, as shown in the figure.

Unfortunately ethical approval for study of our 2015 cohort was only completed in the final week of semester and very few students (in the low single digits) attempted use of the interface. Before this, the MOOC was active but with very few users.

5.6 Statistical Analysis

One of our goals was to create a statistically defensible learner model on the basis of MOOC results. Our intended methodology was to cross-reference MOOC activity against results from the COMP2129 final formative exam.

To this end, we used the learning objectives nominated in our MOOC activity breakdown and assigned these to questions testing similar topics in the MOOC.

This was complicated by the entirely independent authoring of the two sets of material. The exam was authored without evident reference to self-assessment questions in the MOOC; the problem types used are not similar.

The MOOC was not in any way assessable. Assessable components of the course were found in assignments, tutorials and of course in the exam itself; assignments, being primarily long-answer in form, were structured entirely differently from the MOOC, although the underlying material was the same.

We found that the relationship between exam results and MOOC materials was not strong enough to allow consistent conclusions to be drawn from MOOC results in deciding whether particular learning objectives were supported by the MOOC evidence. However, there was sufficient correlation to support some conclusions being made when supporting evidence was particularly strong.

5.6.1 Origin of the Data Set

The data analysed here is collected from the actions of the students who participated in the COMP2129 second-year computer science course at the University of Sydney in 2014, which teaches introductory UNIX and C programming. Students at this point are expected to already be familiar with programming in Java. The SPOC covers the first seven weeks of this course. The second half of the course, tested in the exam but not taught in the SPOC, chiefly concerned multithreaded programming in C under UNIX.

The course in question used a “backwards classroom” format; all students were required to view the videos in the SPOC before attending formally scheduled lectures. Lecture slides were also included in the SPOC course materials. In theory, this should have resulted in the majority of students viewing the majority of video materials over the first seven weeks of the course.
In practice, the student population followed the pattern typical of many MOOCs, with “attendance” dropping significantly over time. While there were ways to view the videos outside of the SPOC (such as directly via YouTube), the Google Analytics results for YouTube views also showed a rapid dropoff over time.

However, there was a brief spike of activity immediately before the final exam, believed to indicate revision by students.

Data was also collected for the corresponding 2015 student cohort. However, the format in 2015 was changed to a traditional lecture series; SPOC participation was suggested and encouraged rather than required. As a result, 2015 student activity was approximately half a percent (0.5%) of that in 2014.

304 students had marks recorded for the final exam in 2014. Of these, 22 had a zero mark recorded (including eight with no recorded SPOC activity); 109 in total (including the aforementioned 8) viewed no videos and completed no problem sets.

All exam questions were manually coded for learning outcomes derived from those created while coding learning outcomes for the SPOC. As parts of the exam concerned material in the latter part of the course which was not covered by the SPOC, some questions could not be coded for learning objectives, although the relevant ACM learning objectives were coded where relevant.

The final exam consisted of 32 multiple-choice questions (including a couple where multiple selections were required) and five long-answer questions, testing material from both parts of the course. Appendix B outlines the learning objectives tested by each question in the exam.

5.6.2 Description of the Data Set

We examined the data with regard to several basic hypotheses. Table 5.1 describes the dependent and independent variables used in this analysis.

The numbered entries here may require some additional explanation. Let us consider the following scenario for a given student, Alice:

- Exam question 4 tests learning objectives A, B, C and D. Alice answered this exam question correctly.
- Alice viewed video X teaching objective C (100 events), video Y teaching objective D (200 events) and video Z teaching objective F (100 events).
- SPOC problem set 12 tests objectives A and B. This student answered 8 questions from this set with 6 correct.
- Problem set 15 tests learning objective D; 2 attempts, 2 correct.
- Problem set 30 tests learning objective E; 7 attempts, 5 correct.

For Alice the values of these variables will then be:
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>Constant component</td>
</tr>
<tr>
<td>$M_n$</td>
<td>Exam mark (as a percentage) for question $n$ in the exam.</td>
</tr>
<tr>
<td>$M$</td>
<td>Overall exam mark (as a percentage)</td>
</tr>
<tr>
<td>video_count</td>
<td>Number of SPOC video events per student. Video events include seeking activity and checkpoints through each video view.</td>
</tr>
<tr>
<td>problem_count</td>
<td>Number of SPOC problem events (including both correct and incorrect) per student</td>
</tr>
<tr>
<td>video_flag$n$</td>
<td>1 if the student viewed SPOC video material teaching learning objectives tested by question $n$ in the exam. 0 otherwise.</td>
</tr>
<tr>
<td>video_flag</td>
<td>Values for video_flag$n$ across all valid $n$.</td>
</tr>
<tr>
<td>problem_flag$n$</td>
<td>1 if the student attempted a SPOC problem teaching learning objectives tested by question $n$ in the exam. 0 otherwise.</td>
</tr>
<tr>
<td>problem_flag</td>
<td>Values for problem_flag$n$ across all valid $n$.</td>
</tr>
<tr>
<td>problem_percent$n$</td>
<td>Overall percentage of correct attempts at SPOC problems teaching learning objectives tested by question $n$ in the exam.</td>
</tr>
<tr>
<td>problem_percent</td>
<td>Values for problem_percent$n$ across all valid $n$.</td>
</tr>
<tr>
<td>play_video</td>
<td>Total number of video “play” events per user</td>
</tr>
<tr>
<td>seek_video</td>
<td>Total number of video “seek” events per user</td>
</tr>
<tr>
<td>problem_check</td>
<td>Total number of short-answer SPOC problem attempts per user</td>
</tr>
<tr>
<td>problem_correct</td>
<td>Total number of correctly answered SPOC problem attempts per user</td>
</tr>
</tbody>
</table>

Table 5.1: Modelling exam results per question vs video and problem flags

- $M_4$ is 100 (percent), since the question was answered correctly.
- video_flag_4 is 1, since Alice viewed at least one video concerning a learning objective taught by exam question 4.
- video_count_4 is 300, since Alice viewed 300 video events concerning learning objectives taught by question 4 (100 for C, 200 for Y.)
- problem_flag_4 is 1, since Alice performed at least one problem concerning objectives tested in exam question 4.
- problem_count_4 is 10, since Alice made ten attempts at problems testing learning objectives tested by exam question 4.
- problem_percent_4 is 80 (percent), since Alice got 8 (2 + 6) correct answers out of 10 (2 + 8) for problems testing common learning objectives.
We now consider several hypotheses with regard to our 2014 COMP2129 data set.

5.6.3 Hypothesis: Activity in SPOC results in improved Learning

Our initial hypothesis was that any activity in the SPOC would improve student learning outcomes. Watching videos would have a small positive effect and performing problem exercises would have a somewhat larger positive effect. Students who did not participate in the MOOC would do worse overall.

While we hoped that activity levels would be correlated with the final exam mark, this proved not to be the case. When the final exam mark ($M$) was correlated against the number of problem attempt events ($\text{problem\_events}$) and the number of video events ($\text{video\_events}$), the result was found to have no appreciable statistical significance (p-value of 0.19 and 0.23 respectively). This held when video events were broken down by event type ($\text{play\_video}$ and $\text{seek\_video}$ with p-values of 0.738 and 0.650 respectively; $\text{problem\_events}$ with a p-value of 0.229).

Figure 5.4 is a scatter plot of exam mark vs. video event counts, and Figure 5.5 shows the same analysis for problem attempts. The graph reflects the analysis; there is no visible correlation.
Figure 5.5: Scatter plot of exam mark vs Problem Solution Events
5.6.4 Hypothesis: Assessment in SPOC indicates improved Learning

However, when the number of correct solutions to problems is factored into the analysis (the problem_correct variable) the least-squares regression is not only highly significant but also permits problem_check to be included as a highly significant variable with a negative coefficient. This tells us that simply doing problems does not contribute to a higher exam mark; the student must also, on the whole, get them correct. Table 5.2 illustrates this.

Note that EdX records some problem_check events with no result (correct or incorrect) included. This analysis only includes those entries where a result is recorded. The “extra” events appear to indicate cases where a problem set was submitted with some individual problems unanswered.

The dependent variable here is the student’s mark in the final semester exam (M).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>50.54342</td>
<td>1.81834</td>
<td>&lt;2x10^{-16}</td>
</tr>
<tr>
<td>problem_check</td>
<td>-0.11248</td>
<td>0.02089</td>
<td>1.47x10^{-7}</td>
</tr>
<tr>
<td>problem_correct</td>
<td>0.34429</td>
<td>0.05534</td>
<td>1.66x10^{-9}</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td>23.19</td>
<td>4.639x10^{-9}</td>
</tr>
</tbody>
</table>

Table 5.2: Modelling exam results vs problems attempted and problems correct

This correlation indicates that every problem attempted within the MOOC would, on average, indicate approximately a 0.22% improvement in the student’s final exam mark if the answer is correct, or a 0.11% drop in the final exam mark if the problem was attempted but incorrect.

However, the evidence is insufficient to conclude that attempting questions in the SPOC results in better exam marks. Furthermore, the model’s standard error is quite high as 23.19, and so has little predictive power.

Figure 5.6 illustrates the relation between performing problems within the SPOC and final exam mark. The line indicates the least-regression fitted correlation.

5.6.5 Hypothesis: Correlation is higher when variables have a larger number of possible values

We observed that in all models, the standard error of the model was unacceptably high. Furthermore, the accuracy of the model appeared to increase with the size of the range covered by that variable: a variable with a larger range would, all things being equal, correlate more closely with any other variable than a second variable with a smaller range of possible values.
This would mean that multiple choice questions would generally correlate poorly and long-answer questions would correlate well.

To confirm this, we examined cross-correlations between multiple-choice questions in the exam, compared to multiple-choice questions in the MOOC, compared to long-answer questions in the MOOC.

We limited our analysis to the single topic which was best represented across the exam (three multiple choice questions and one long-answer question) and the MOOC (seven long-answer questions). We then generated a matrix of coefficients of correlation between each variable.

The variables used were: MC#1, MC#2 and MC#3, being the individual correctness scores for the three relevant multiple-choice questions in the exam; “MC total”, being the total of these three scores; “Long Answer”, being the total for the exam’s long-answer question on strings; “MOOC%”, being the percentage of correct answers in the MOOC; and Exam Total, being the student’s total score in the exam.

The correlation between “MC total” and the individual multiple-choice questions is high, as expected, because these are not independent variables.

You will notice that the correlation between the exam multiple choice result and the exam long-answer question result (0.21) is actually lower than the
Table 5.3: Matrix of coefficients of correlation

correlation between the MOOC percentage and both the correlation between MOOC and exam long-answer (0.332) and exam multiple-choice (0.335). The MOOC percentage also correlates better against the overall exam result than the exam multiple-choice total for these three questions (Although the long-answer correlation is still better).

Correlation between separate multiple-choice exam questions is exceptionally poor, below the 0.1 correlation level, although the first question by itself has a coefficient of 0.274, better than the total of the three questions.

In general this appears to support our hypothesis that the issue here is not one of the multiple choice questions in the exam correlating poorly against the multiple choice questions in the MOOC, but that they do not correlate particularly well with anything.

Fundamentally, the exam does not provide enough depth of data for any single learning objective to be able to make predictions about student knowledge concerning these same learning objectives in the MOOC. The same basic problem occurs within the exam; the problem is not that the exam data is poor, but that there is not enough of it to model the learner objectives accurately.

5.6.6 Other possible predictive hypotheses

The learning objective which is most prevalent in the exam is “Strings”, with three multiple-choice questions and much of one long-answer question dedicated to this topic. To limit duplication of work, the focused on this one learning objective to determine how an improved statistical basis for determining our learning objectives might be determined.

First we consider the hypothesis that skilled students will answer questions correctly more consistently on their first attempt. We create a variable (mooc_first_maxrun) giving the maximum size of the run of correct answers. Observe figure

The model standard error for this regression is only 16.41, and all variables are highly significant. Its predictive value is significantly improved.

Let us next attempt a simple regression of percentage marks of the same learning objective between MOOC and exam.
Figure 5.7: Scatter plot of exam mark vs Length of Run of Initial Correct Answers for strings
Figure 5.8: Scatter plot of exam mark for Strings LO vs percentage of MOOC Strings attempts correct
The model error in this case is significantly worse, at 4.426 out of a maximum total of 19 marks, for an overall error of 23%.

Clearly returning to our original method is not particularly helpful.

5.6.7 Breaking data down by Decile

We must recall here that our primary purpose in this analysis is to populate our learner model with data from the MOOC. Essentially, we seek true positives; as the student is free to correct the model, we want to avoid false positives, but some false negatives are acceptable.

We compare the scatter plots for the second and ninth deciles. The mean is clearly higher: 14.8 for the exam LO total and 70.87% for the MOOC percentage, compared to 8.1 and 40.9 for the second decile. Figure 5.9 gives results for the second decile and Figure 5.10 gives results for the ninth decile.

Overall we find that for this data the model residual remains at around 25% of the overall mark if our choice of learning objectives is done with due care.

Several other possible metrics for prediction of exam LO results were also tested. We found that the best of them had about as much variability as the overall MOOC percentage per-LO; given that the percentage is simple to calculate we have chosen it as our preferred metric. While it does have significant
Figure 5.10: Decile 9: Scatter plot of exam mark for Strings LO vs percentage of MOOC Strings attempts correct
problems with variability, those problems are also found in other data.

Some of the independent variables examined included:

- The number of questions answered correctly on first attempt. (Hypothesis: a student knowing the material will correctly answer each question on their first attempt.)
- A flag value for 2, 3 or 4 sequential correct answers. (Hypothesis: Once the topic is understood, further answers should all be correct.)
- The number of questions attempted irrespective of answers being correct. (Hypothesis: students learn by attempting problems, so a student who attempts more problems should learn more.)
- Student answers for entirely unrelated questions in the exam. (Hypothesis: learning should be uniform across all topic areas and the exact learning objective should not matter.)
- Student answers for related questions in the exam. (Hypothesis: students familiar with one topic but not others will answer more consistently for that topic.)

All of these variables correlated positively, but none correlated significantly better than the statistic chosen (the percentage of correct answers.) In general, it was found that statistics with less discreteness (a greater range of possible values) correlated more highly than those with a smaller number of discrete possible values.

With some outliers, the difference between higher-scoring students and lower-scoring students was clear from the results, but the variability of these results was too high for them to have significant predictive value, save in identifying the highest-achieving students.

5.6.8 Statistical Result for Personis Resolver

For the COMP2129 course the “bare pass” mark is 40% and our results show that the MOOC question percentage is close to the exam percentage. If we construct an error bar with mean at 40% and standard deviation of 25%, a standard probability fit at the 90% certainty level should allow us to make a positive conclusion.

This may be inaccurate for more complex topics, but we would expect less capable students to do worse on such questions rather than better, so our result should hold.

In fact, other results not detailed here indicate that the slope coefficient for more complex topics is consistently steeper, so the odds of a false positive should be significantly reduced.
Chapter 6

Evaluation

Here we demonstrate through a user study how our OLM can be used to answer the questions posed in the Introduction as well as selected other questions of interest. Chapters 3-5 demonstrated the design, construction and population of our OLM; it remains to show that the OLM is useful in answering our questions.

These questions represent the hypotheses established in Chapter 1, as summarised in Table 6.1.

6.1 Study Design

We conducted a small study (five individuals) using the “Thinkaloud” protocol described by Clayton Lewis [Lewis, 1982] in order to gain rich qualitative feedback on the quality of the interface and system.

Originally this study was proposed for members of the COMP2129 class itself, but as ethics approval was delayed until after all coursework was complete, this proved to be impractical. With the approval of the Ethics committee, participants were instead chosen from people who had participated in COMP2129 or a similar course in the past.

The study was approved by the University of Sydney Human Research Ethics Committee as Project Number 2015/277 on June 6th, 2015, with emendations to the protocol, primarily concerning approved participants, approved on October 7th, 2015. A copy of the Participant Information Statement may be found in Appendix G.

The directives in the study were designed to test each hypothesis at least once and each key hypothesis at least twice. The study was broken into two parts (first person, “my results” and third person, “their results”) to examine use of these hypotheses in different contexts.
<table>
<thead>
<tr>
<th>Hyp. #</th>
<th>Summary label</th>
<th>Hypothesis text</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>Overview</td>
<td>That the system usefully shows overall progress of this student on the learning activities</td>
</tr>
<tr>
<td>H2</td>
<td>On-track</td>
<td>That the system usefully shows in which learning objectives the learner has met the teacher's expectations</td>
</tr>
<tr>
<td>H3</td>
<td>Behind</td>
<td>That the system usefully shows those learning objectives where the student is lagging behind expectations</td>
</tr>
<tr>
<td>H4</td>
<td>Activity-Type-Progress</td>
<td>That the system enables answering H1-3 for a particular class of activity (video, exercise, discussions... )</td>
</tr>
<tr>
<td>H5</td>
<td>Act</td>
<td>That the system assists the student in finding learning resources associated with any given learning outcome</td>
</tr>
<tr>
<td>H6</td>
<td>Openness</td>
<td>That the system permits the student to view and modify their Learner Model</td>
</tr>
<tr>
<td>H7</td>
<td>Selectivity</td>
<td>That the system allows the learner to view parts of their model selectively</td>
</tr>
<tr>
<td>H8</td>
<td>Topology Mapping</td>
<td>That the system permits the student to view different logical topologies for the organisation of course material</td>
</tr>
<tr>
<td>H9</td>
<td>Comparison</td>
<td>That the system permits comparison between multiple learner models in a useful fashion</td>
</tr>
<tr>
<td>H10</td>
<td>Curriculum</td>
<td>That the system permits cross-comparison of selected portions of the curriculum</td>
</tr>
</tbody>
</table>

Table 6.1: Thesis Hypotheses

6.2 Study Structure

The study was broken into two nominal sections, the first from the point of view of a pseudonymous student (“Alice”) and the second from the point of view of a tutor for Alice and, primarily, a second student (“Bob”).

The first part of the study uses a Curriculum Reference Model. The second part uses a User Reference Model. The “ideal”/“typical” models are not used for this study as the skill sets involved in driving the user interface are essentially the same as for a Curriculum Reference Model, and we wished to avoid prolonging the study unnecessarily.

The second part of the study includes comparison against a student whose model is “empty” (nothing shown as known) using the standard Resolver as the student involved limited themselves primarily to the videos, which the system, by default, regards as insufficient evidence of learning. Use of a resolver with a more optimistic interpretation of the evidence, as directed in the study, shows this same model as being moderately well populated.

This was done to highlight the importance of awareness of the method being used to resolve values for the learner model. Participants were directed to leave the resolver at its second (non-default) setting for the remainder of the study.
The “bob” user, while chosen at random from users who scored lower in the exam than “alice”, had one interesting characteristic which lent some complexity to their associated tasks. The “bob” user completely ignored all assessment tasks. Since video results were ignored in the default resolver due to low statistical correlation against exam results, this meant that the default view for “bob” appears blank.

Participants were presented with a set of tasks to be performed with the user interface; the full text of the thinkaloud may be found in Appendix E.

We list how these tasks correspond to our hypotheses in the detailed analysis of each question, then examine the validity of each individual hypothesis in our post-analysis.

Hypothesis 8 (H8) could not be tested for linking between separate MOOCs as only a single MOOC was studied. However, as the filtering option use a variation on the same concept, those questions that test filtering are listed as testing H8.

The thinkaloud participant pool comprised five participants, three male and two female. All were postgraduate computer science students.

Each question description includes a table showing the difficulty encountered by each participant in answering the question. This is encoded as one of three characters. O indicates that little or no coaching was required and the user had no trouble answering the question. X indicates that detailed explanation or coaching was needed. = indicates that an intermediate level of coaching was required or that the user independently found the answer with some difficulty.

Table 6.2 breaks down which hypotheses are tested by which questions in the study.

<table>
<thead>
<tr>
<th>Question</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>H4</th>
<th>H5</th>
<th>H6</th>
<th>H7</th>
<th>H8</th>
<th>H9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>1.10</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.2: Hypotheses Tested by each question
6.3 Thinkaloud Study - Outline of Participants

The terms of the study required that all participants know the material covered by the MOOC. Table 6.3 outlines some basic information about the participants.

Two of the five participants were female; all were between 20-30 years old, from a wide range of ethnicities. Several had experience in tutoring undergraduate computer science, a background considered useful in addressing the second part of the study.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Source of Experience</th>
<th>Tutoring Experience?</th>
<th>Time Taken (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>COMP2129</td>
<td>Yes</td>
<td>32</td>
</tr>
<tr>
<td>P2</td>
<td>COMP2129</td>
<td>Yes</td>
<td>63</td>
</tr>
<tr>
<td>P3</td>
<td>Similar Course</td>
<td>Yes</td>
<td>63</td>
</tr>
<tr>
<td>P4</td>
<td>COMP2129</td>
<td>No</td>
<td>71</td>
</tr>
<tr>
<td>P5</td>
<td>COMP2129</td>
<td>No</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 6.3: Information about participants

We now proceed with an analysis of each individual question, followed by a discussion of hypotheses across all questions.

6.4 Thinkaloud Study - Detail of each Question

Here we consider participant response for each question.

Question 1.1

"Please log in using the provided username and login. What does "Alice" know well in the COMP2129 course? Give your initial impressions."

Hypotheses: H1

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>=</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.4: Ease of understanding - Question 1.1

Here the user fills in the user name and password details for "alice" at top left while the “Simple” view is selected, as shown in Figure 6.1.

Two users (P2 and P3) had some difficulty working out how "login" worked with this interface, given that there is no explicit "login" button. (Login is handled by intercepting change events on the login fields.) P2 was expecting the login prompt to be in a corner rather than incorporated into the larger panel:

P2: That is not obvious! That is not obvious at all!
Once login was successful, participants had no trouble interpreting the model overview, although some were briefly confused about whether “known” items were represented by black dots or white.

**P5:** *Alice is not doing very well in this course because she doesn’t have a lot of black dots on the model, I guess. Yeah, she’s better in programming languages but not very well in operating systems, I guess.*

**Question 1.2**

“Determine the meaning of the red text.”

_Hypotheses:_ None (interface usability)

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 6.5: Ease of understanding - Question 1.2

The red text indicates the element that the system thinks would be best used as a “next thing to learn”. Unfortunately the significance of this is described deep in the help and it is not immediately intuitive from the default interface. Figure 6.2 shows the result of selecting the Help button, which explains this functionality, but all participants required guidance to find this.

We conclude that this interface element is essentially a failure and requires considerable re-working. It does serve an important function, but this probably
Figure 6.2: Sample Solution: Question 1.2

This shows "learning outcomes" (things that you can learn) in the COMP2129 MOOC for the selected Learner Model. A Learner Model is basically what the system thinks this person knows.

Each small circle is a learning outcome. The meanings of the colours of the circles are given in the colour key. There is a more detailed explanation below.

Detail: Click on these small circles for a detailed explanation and access to teaching materials. Hover over any circle for a brief descriptive tooltip.

Next Learning Outcome: Our suggestion for what to learn next has a thick, dashed/dotted border and red text.

Zoom: Click a large circle to zoom in. Click outside of the zoomed circle to zoom back out.

The Reference Model is only shown in Expert mode; it is a second learner model or curriculum that you are comparing the Main model against. The Filter causes a specified, limited subset of the model to be shown.

Colours of the learning outcomes are determined by combining the "main" model with the "reference" model. The ACM "Knowledge levels" (indicating how thoroughly material is understood) are ignored for this; the only distinction made is "Unknown" or "Known."

• If they match, or if there is no reference model, the "Known" (true) and "Unknown" (false) colours are used.
• If there is a reference model, the "Exceeds Expectations" colour is used if the "Main" model is known but the "Reference" model is unknown.
• If the "Fails to Meet Expectations" colour is used when the Main model is unknown but the Reference model is Known.

The larger circles represent nested portions of the learner model - for example, "Operating Systems" falls under "Computer Science."

If you search for text, any items with that text in the description will be shown in blue. The highlighted item with red text is the system's suggestion for your "next" thing to learn.
needs to be made an active function (such as a “Show me what to learn next” button) rather than a passive interface element which chiefly serves to confuse.

**Question 1.3**

“Does the system think "Alice" knows about "for" loops in C?"

**Hypotheses:** H2

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 6.6: Ease of understanding - Question 1.3

Participants used a variety of strategies in answering this question. The population was split down the middle between “searchers” and “model explorers” (with P2 shifting from the first to the second when the result of a search for “for” was too cluttered.) Figure 6.3 indicates the “Search” dialogue used to answer this question.

“Searchers” used the search function at top right while “interface explorers’ explored the model graphically until finding the desired element. P4 used the latter approach without using the “zoom” function of the learner model, which required manual inspection of many hovertext entries, slowing the searching process dramatically.
Figure 6.3: Sample Solution: Question 1.3. The arrows point to the key portions of the interface for answering this question.

**Question 1.4**

“As "Alice", you think that the system is incorrect about her knowledge of "for" loops. Can you correct this?”

*Hypotheses: H6*

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.7: Ease of understanding - Question 1.4

This question was handled fairly well by all participants, since they had clicked on the “for loop” node as part of answering the previous question and the implementation of this is clear and obvious from within that dialogue box. Figure 6.4 shows this dialogue box.

There was some confusion about the effects of making this change, as an interface bug required a forced manual reload of the model to show the change in the displayed model. Figure 6.5 shows a detail of the learner model (with an arrow pointing to the ForLoop component) after the change has been integrated into the learner model.
Figure 6.4: Sample Solution: Question 1.4. The arrow indicates the button to press to indicate that this item is known.

Figure 6.5: Change to Learner Model after 1.4 (detail)
Question 1.5

“Attempt to locate learning resources concerning this topic.”

Hypotheses: H5

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 6.8: Ease of understanding - Question 1.5

This function was also found very quickly as it is located within the detailed component view with which participants had previously become familiar. Figure ?? indicates the link to be followed to access the learning resources. The main difficulty encountered here was with the MOOC links from the Learning Resource list (shown in Figure 6.7) as these cannot be used unless the MOOC is already open.

Only P5 had any significant difficulty in finding these learning resources as the other participants had noted the learning resource link while answering Question 1.4. P3 did need an explanation of what comprises a “learning resource.” We take this as a reminder that not all users will be familiar with educational jargon; it may be necessary to rephrase some interface elements to reflect this.

Three of our users attempted to click on the link in the learning resources box. P2 tried the MOOC link then the YouTube link; the other two tried only the YouTube link. Two of those who selected the YouTube link expressed pleasant surprise that the link jumped directly to the relevant portion of the video:

**P1:** So I can go watch the YouTube - Oh, it’s gone to the right bit in the YouTube video. That’s cool.
Figure 6.6: Sample Solution: Question 1.5. The arrow points to the link for additional learning resources.

Figure 6.7: List of Learning Resources displayed for Q1.5.
Question 1.6

“If you have not already done so, switch the interface from “Simple” to “Expert” mode.”

Hypotheses: None (interface detail)

This question required switching from the Simple to the Expert interfaces, as compared in Figure 6.8. Most participants managed this without too many problems, but two missed the radio buttons for making the switch and instead selected the “Complex” radio button on the panel below.

P2: You know what - until I saw this question I did not realise that there were two modes.

One option for addressing this issue is to change the user interface presentation to a more context-sensitive layout as is used for colours. Rather than having the Simple/Expert selector at the top, have the “Custom” selection immediately below this function.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.9: Ease of understanding - Question 1.6
Question 1.7

“See how well Alice has covered material in self-assessment exercises in the MOOC.”

Hypotheses: H4

The common mistakes made in 1.7 were either selecting the Self-Assessment exercises as the Main model, or searching for Assessment under Filters. Figure 6.9 indicates the key interface elements.

One interesting aspect of this was that participants were fairly strict about what they were looking for - they were looking for phrasing to match exactly what the question specified and signalled confusion when the exact text was not present in the interface. Wording of selected menu elements was adjusted in response to this, with references to “MOOC exercises” changes to “Self-Assessment Exercises”.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.10: Ease of understanding - Question 1.7
Figure 6.9: Sample Solution: Question 1.7. Arrows indicate where changes must be made to satisfy this exercise.

**Question 1.8**

“Experiment with the colour selections. Explain in your own words what you think the colours mean, using specific examples of what Alice knows vs. what was covered in self-assessment exercises.”

*Hypotheses: H9*

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.11: Ease of understanding - Question 1.8

We omit the visual figure in this instance as the procedure and “solution” for this question was, by the nature of the question, highly variable. The goal of this question was to encourage participants to consider and explore the exact meaning of the colour map, in particular how overlapping truth values (the “exceeds” and “fails” colour selections) should be interpreted.

The most common variations chosen were green for True, and swapping black and white. P2 at one point asked about restoring colours to their original values, having missed the button already present for this function.

One participant complained that while the explanations went into some effort to explain what the colours were in reference to the Reference Model, the nature of the Reference Model was not itself explained.

**P1**: So what’s a reference model compared to a main model? Does it, is it explained to you? I don’t think it is.
On the other hand, P2 seemed to understand how colours were chosen with relatively little explanation, and when confused about how truth and false values were calculated, found the system’s explanation fairly easily: **P2: It does tell me the rules if I click into the bubbles. Excellent.**

**Question 1.9**

"Try some of the alternatives listed under Visualisations. (Note that these are prototypes with only partial functionality.) Pick one of these and compare it with the default layout."

**Hypotheses:** None (Comparison of visualisations)

The question was posed to generate feedback about the visualisation compared to some simple alternatives. These alternatives are shown in Figures 6.10, 6.11 and 6.12. Three of the five participants preferred the Dendrogram layout, one (P4) preferred the Sunflower visualisation and one preferred the circular treemap (P5).

P2 noted that there were features of the primary visualisation that he had not noticed until seeing the alternative visualisations.

We believe that our participants’ reaction indicates that having multiple visualisations available is advantageous. The general preference for “traditional” tree layouts suggests that such a layout should be included in any such compilation.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.12: Ease of understanding - Question 1.9
Figure 6.10: Question 1.9 First Visualisation: Circular Treemap with slanted text

Figure 6.11: Question 1.9 Second Visualisation: Sunflower
Question 1.10

“Please comment on the actual learner model structure. Is it too big? Too small? Are parts confusing?”

Hypotheses: H1, H7

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.13: Ease of understanding - Question 1.10

P1 liked the openness of the whole model.

**P1:** I like the idea that it’s all there. That’s good. I’m happy that things aren’t hidden from me.

P3 liked the standard visualisation as an overview:

**P3:** I really like to see the overall view of the close result when I see the bubble chart… is very good. … this one [the dendrite view] is easy to access and easy to see the endpoint.

Question 1 Overall Commentary

“Having explored the interface thoroughly, give your overall impressions, including any criticisms you may have.”

Hypotheses: None/All

P2 complained about the T1/T2 structuring used for selected portions of the learner model. This structuring was inherited from the ACM curriculum but hiding it by using a filter may have improved user comprehension.
Question 2.1

“Log into “bob”’s model and evaluate the usefulness of the view from the perspective of a tutor.”

Hypotheses: H1

Several participants had trouble logging in as “bob”, but in all cases this was because Bob’s password includes a “I” (eye) which, with a sans-serif font, was easily confused with a lower-case L. Discounting this issue, no serious problems were encountered at this point.

None of the problems encountered in 1.1 were encountered for 2.1. Figure 6.13 shows an example where bob has logged in.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.14: Ease of understanding - Question 2.1

Question 2.2

“Describe “bob”’s progress on the course. Is there a way to vary the view to show a more “optimistic” interpretation of the evidence? Indicate whether you believe the more optimistic view is preferable and explain your opinion.”

Hypotheses: H2,H3

Bob’s default model resolves to “nothing found” with the default Resolver. This question invites the student to search for a different way of resolving the
evidence (i.e. a different Resolver.) It relates to H3 because, with the default resolver, “bob” apparently knows nothing; participants were thus forced to explore why the model showed such a lack of progress.

Some minor prompting was required for this problem to direct participants to the Resolvers option. This indicates that this needs to either be renamed, or for explanatory text to be added.

As indicated by our summary, all users here required significant explanation concerning exactly what the option actually does. P1 had less trouble due to some prior familiarity with Personis. Figure 6.14 indicates the interface selection for changing the Resolver and shows Bob’s model using the more optimistic resolver selection.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 6.15: Ease of understanding - Question 2.2

Question 2.3

“Evaluate “bob”’s progress against the learning objectives tested in the final exam. Describe what the different colours mean in your own words, and your opinion of how they are used.”

Hypotheses: H3, H4, H8, H9

This question involved selection an alternative Resolver (evidence interpretation). P1 pointed out that for students a more optimistic default may be preferable to prevent them from being discouraged:
P1: All right, so it said explain, like, if that's preferable, um, I think if you were trying to not make them feel horrible, probably, ah, let's see if I can refer it against something more optimistic.

P2 felt that calling the models the Main and Reference models was confusing:

P2: In that case why not call them Model 1 and Model 2 and not have these explanations?

H3 and H4 are relevant here because the question examines Bob’s lack of performance compared to the final exam. H8 and H9 are relevant because the solution can use either a Reference Model for the relevant portion of the curriculum, or a Filter limiting the part of the model to be used. Figure 6.15 shows the Reference Model approach.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>—</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 6.16: Ease of understanding - Question 2.3

**Question 2.4**

“See how much of the final exam was known by “bob” as of the end of March.”

**Hypotheses:** H1, H4, H8

Question 2.4 required integrating use of the Resolver (as selected in 2.2) with the exam objectives (2.3) and adding a date filter (an as-yet unused element). Of these, the Date filter was the only element requiring change. Figure 6.16
indicates the relevant interface element.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O =</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.17: Ease of understanding - Question 2.4

**Question 2.5**

"Compare how much had been learnt by Alice and Bob by the end of term."

**Hypotheses:** H9

This question introduces the concept of a second student as a reference model. All but one participant answered this fairly quickly, with that one entering Alice’s details as the Main model while leaving the Reference Model blank. This was quickly corrected once pointed out. There was some minor confusion around ‘... by the end of term’ which simply required removal of the Date filter. Figure 6.17 shows those elements requiring change.

At this point the study participants seemed to be becoming quite comfortable with the concept of comparing Learner Models.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>O =</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.18: Ease of understanding - Question 2.5
Question 2.6

“Compare how much had been learnt by both solely within the topics tested by the exam.”

Hypotheses: H4, H8, H9

This question added the need to use a Filter for exam topics with the two students as Main and Reference Models. Three of our students initially tried to select the Exam Assessment as a Reference Model before realising that this removed Alice from the view.

The strategies used by our participants at this point showed a thorough familiarity with use of Reference Models, but use of the Filter capabilities of the interface was less intuitive. As the previous uses of Filters could also be answered with a Reference Model, this is perhaps unsurprising.

Figure 6.18 indicates the Filter element which required modification to answer this question.

<table>
<thead>
<tr>
<th>Participant</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of understanding</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
</tbody>
</table>

Table 6.19: Ease of understanding - Question 2.6

Activity when dealing with this question showed perhaps an over-reliance on the use of Reference Models. The first impulse of each participant was to set the Reference Model to the exam topic set. This eliminated “bob” from the results. Three of the five participants realised this before actually changing the reference model. All participants required coaching to use the “Filter” function
to vary the context of the learner model comparison.

This suggests that the Reference Model (H9) was understood thoroughly at this point but that the Filter (H8, H10) had been relegated to background status and essentially disregarded.

User comments following the study

After the questions in the study were answered, all users were asked for any additional comments and whether they believed the tool used in the study would be useful in a real-world environment.

Our participants were uniformly positive about the ability to compare learner models, although several repeated their preference for the Dendrite Tree visualisation.

**P1:** I got the hang of it I think at the end. Um, it was interesting being able to compare students to each other. I think I’d be interested to do that, as a tutor. Yeah, just to get like a gauge - cos sometimes it’s difficult to tell whether the most outspoken student is the smartest student... in your class. And so yes, I’d be interested to be able to compare students like that.

**P2:** I found it a little confusing to be honest... Maybe with a few user interface tweaks. Sometimes when I change things and it reloads and I’m not really sure what has changed. With the comparing one, it would be nice to have two, side by side. I thought it was interesting overall. I think if a teacher or a tutor was teaching a course... actually I wouldn’t use the bubble model. I didn’t like the bubble. I think if they were using the dendrogram and they were sitting with the student, I think it would help the student quite a lot.
6.5 Outline of Results

Table 6.20 summarises the difficulty encountered by each participant across all questions.

Table 6.21 tabulates the approximate time (in seconds) required for each participant to complete the study, as well as the amount of coaching or assistance supplied by the supervisor (also in seconds).

It is vital when examining this table to realise that a thinkaloud study is inherently qualitative, not quantitative in nature. Having the participant spending a large amount of time on a problem is not an inherently bad result, because that time may well have been spent in exploring the interface rather than struggling to understand it.

Time spent in each task is primarily included as a primitive metric for the complexity of each sub-task.

Where this table is chiefly of interest is in the “Coaching” figures, which represent the time spent by the experimenter in explaining the interface to the participant. If no coaching is required, the interface is easy to understand.

<table>
<thead>
<tr>
<th>Question</th>
<th>Hypotheses</th>
<th>Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>P1</td>
</tr>
<tr>
<td>1.1</td>
<td>H1</td>
<td>O</td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td>=</td>
</tr>
<tr>
<td>1.3</td>
<td>H2</td>
<td>O</td>
</tr>
<tr>
<td>1.4</td>
<td>H6</td>
<td>O</td>
</tr>
<tr>
<td>1.5</td>
<td>H5</td>
<td>O</td>
</tr>
<tr>
<td>1.6</td>
<td></td>
<td>=</td>
</tr>
<tr>
<td>1.7</td>
<td>H4</td>
<td>=</td>
</tr>
<tr>
<td>1.8</td>
<td>H9</td>
<td>=</td>
</tr>
<tr>
<td>1.9</td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>1.10</td>
<td>H1,H7</td>
<td>O</td>
</tr>
<tr>
<td>2.1</td>
<td>H1</td>
<td>O</td>
</tr>
<tr>
<td>2.2</td>
<td>H2,H3</td>
<td>=</td>
</tr>
<tr>
<td>2.3</td>
<td>H3,H4,H8,H9</td>
<td>=</td>
</tr>
<tr>
<td>2.4</td>
<td>H1,H4,H8</td>
<td>O</td>
</tr>
<tr>
<td>2.5</td>
<td>H9</td>
<td>O</td>
</tr>
<tr>
<td>2.6</td>
<td>H4,H8,H9</td>
<td>=</td>
</tr>
</tbody>
</table>

Table 6.20: Summary of Results Across all Questions
<table>
<thead>
<tr>
<th>Qn #</th>
<th>Hyp. Tested</th>
<th>P1 Time</th>
<th>P1 Coach</th>
<th>P2 Time</th>
<th>P2 Coach</th>
<th>P3 Time</th>
<th>P3 Coach</th>
<th>P4 Time</th>
<th>P4 Coach</th>
<th>P5 Time</th>
<th>P5 Coach</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>H1</td>
<td>130</td>
<td>16</td>
<td>234</td>
<td>4</td>
<td>258</td>
<td>32</td>
<td>69</td>
<td>11</td>
<td>225</td>
<td>9</td>
</tr>
<tr>
<td>1.2</td>
<td>-</td>
<td>78</td>
<td>3</td>
<td>302</td>
<td>30</td>
<td>113</td>
<td>3</td>
<td>407</td>
<td>22</td>
<td>295</td>
<td>37</td>
</tr>
<tr>
<td>1.3</td>
<td>H2</td>
<td>70</td>
<td>2</td>
<td>86</td>
<td>2</td>
<td>62</td>
<td>1</td>
<td>210</td>
<td>22</td>
<td>96</td>
<td>18</td>
</tr>
<tr>
<td>1.4</td>
<td>H6</td>
<td>33</td>
<td>6</td>
<td>81</td>
<td>8</td>
<td>106</td>
<td>19</td>
<td>65</td>
<td>10</td>
<td>112</td>
<td>15</td>
</tr>
<tr>
<td>1.5</td>
<td>H5</td>
<td>45</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>113</td>
<td>7</td>
<td>60</td>
<td>5</td>
<td>55</td>
<td>17</td>
</tr>
<tr>
<td>1.6</td>
<td>-</td>
<td>55</td>
<td>10</td>
<td>50</td>
<td>0</td>
<td>24</td>
<td>11</td>
<td>71</td>
<td>29</td>
<td>99</td>
<td>11</td>
</tr>
<tr>
<td>1.7</td>
<td>H4</td>
<td>200</td>
<td>25</td>
<td>468</td>
<td>58</td>
<td>75</td>
<td>0</td>
<td>443</td>
<td>55</td>
<td>187</td>
<td>7</td>
</tr>
<tr>
<td>1.8</td>
<td>H9</td>
<td>280</td>
<td>20</td>
<td>139</td>
<td>10</td>
<td>170</td>
<td>12</td>
<td>202</td>
<td>61</td>
<td>235</td>
<td>3</td>
</tr>
<tr>
<td>1.9</td>
<td>-</td>
<td>65</td>
<td>3</td>
<td>170</td>
<td>7</td>
<td>69</td>
<td>11</td>
<td>281</td>
<td>33</td>
<td>356</td>
<td>34</td>
</tr>
<tr>
<td>1.10</td>
<td>H1,H7</td>
<td>32</td>
<td>15</td>
<td>66</td>
<td>4</td>
<td>223</td>
<td>65</td>
<td>166</td>
<td>14</td>
<td>194</td>
<td>16</td>
</tr>
<tr>
<td>1.X</td>
<td>-</td>
<td>34</td>
<td>0</td>
<td>66</td>
<td>4</td>
<td>174</td>
<td>25</td>
<td>32</td>
<td>2</td>
<td>466</td>
<td>27</td>
</tr>
<tr>
<td>2.1</td>
<td>H1,H2</td>
<td>122</td>
<td>6</td>
<td>245</td>
<td>47</td>
<td>281</td>
<td>37</td>
<td>269</td>
<td>23</td>
<td>133</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>H3</td>
<td>128</td>
<td>18</td>
<td>251</td>
<td>44</td>
<td>229</td>
<td>51</td>
<td>296</td>
<td>28</td>
<td>479</td>
<td>101</td>
</tr>
<tr>
<td>2.3</td>
<td>H3,H4,H9</td>
<td>100</td>
<td>2</td>
<td>152</td>
<td>26</td>
<td>242</td>
<td>88</td>
<td>108</td>
<td>5</td>
<td>350</td>
<td>106</td>
</tr>
<tr>
<td>2.4</td>
<td>H1,H4</td>
<td>68</td>
<td>0</td>
<td>93</td>
<td>3</td>
<td>319</td>
<td>32</td>
<td>38</td>
<td>2</td>
<td>177</td>
<td>4</td>
</tr>
<tr>
<td>2.5</td>
<td>H9</td>
<td>138</td>
<td>4</td>
<td>293</td>
<td>26</td>
<td>118</td>
<td>13</td>
<td>465</td>
<td>37</td>
<td>396</td>
<td>55</td>
</tr>
<tr>
<td>2.6</td>
<td>H4,H9</td>
<td>105</td>
<td>15</td>
<td>415</td>
<td>65</td>
<td>390</td>
<td>30</td>
<td>225</td>
<td>24</td>
<td>111</td>
<td>23</td>
</tr>
<tr>
<td>2.X</td>
<td>-</td>
<td>41</td>
<td>0</td>
<td>186</td>
<td>44</td>
<td>197</td>
<td>26</td>
<td>285</td>
<td>120</td>
<td>223</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>Total Time</td>
<td>1724</td>
<td>145</td>
<td>3352</td>
<td>382</td>
<td>3163</td>
<td>463</td>
<td>3692</td>
<td>503</td>
<td>4189</td>
<td>626</td>
</tr>
<tr>
<td></td>
<td>Percent Coaching</td>
<td>7.8%</td>
<td>10.2%</td>
<td>12.8%</td>
<td>12.0%</td>
<td>13.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.21: Time taken by Thinkaloud Participants

One interesting point of difference here is the time taken by the first participant compared to other users, at only around half the time taken by the next-fastest participant. We observed that this participant started out by attempting to explore the interface rather than attempting to answer the study questions and hypothesise that initial exploration period was useful when answering later questions. Later participants were guided to proceed with the

1 included some time on free exploration of the interface.
2 Initially confused by lack of an explicit "login" button.
3 Included some time playing with colour selections as required for 1.8.
4 Was confused by differences between the wording of the question and that used by the interface ("MOOC exercises" vs "self-assessment exercises")
5 Several participants initially misparsed the introduction to section 2 and initially assumed that "bob" was a tutor for alice, rather than the questions being posed to a tutor for two students.
6 Several participants had trouble with this question as the purpose of the "Resolver" control is not immediately obvious.
7 There was an error in the text of this question when this participant sat the study which required verbal correction.
8 P2 initially through that the system was comparing alice and bob independently to a third model and did not understand how the model comparison was being handled until the interviewer explained this explicitly.
study questions at an earlier time and this may have impacted negatively on their understanding of the interface.

P1 not only finished quickly, but required the least amount of supervisor coaching both on an absolute and relative basis.

If we take these results and break them down according to hypotheses tested we see some interesting patterns. Consider the breakdown shown in Table 6.22.

<table>
<thead>
<tr>
<th>Hyp. Description</th>
<th>Hyp.</th>
<th>Tested by</th>
<th>Time</th>
<th>Coach</th>
<th>Coach %</th>
<th>Time/#Qn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>H1</td>
<td>1.1,1.10,2.1,2.4</td>
<td>2158</td>
<td>215</td>
<td>9%</td>
<td>540</td>
</tr>
<tr>
<td>On-track</td>
<td>H2</td>
<td>1.3,2.2,2.4</td>
<td>524</td>
<td>45</td>
<td>8%</td>
<td>175</td>
</tr>
<tr>
<td>Behind</td>
<td>H3</td>
<td>2.2,2.3</td>
<td>2147</td>
<td>446</td>
<td>17%</td>
<td>1074</td>
</tr>
<tr>
<td>Activity-type-progress</td>
<td>H4</td>
<td>1.7,2.3,2.4,2.6</td>
<td>4266</td>
<td>425</td>
<td>9%</td>
<td>1067</td>
</tr>
<tr>
<td>Act</td>
<td>H5</td>
<td>1.5</td>
<td>328</td>
<td>29</td>
<td>8%</td>
<td>328</td>
</tr>
<tr>
<td>Openness</td>
<td>H6</td>
<td>1.4</td>
<td>397</td>
<td>58</td>
<td>13%</td>
<td>397</td>
</tr>
<tr>
<td>Selectivity</td>
<td>H7</td>
<td>1.10</td>
<td>547</td>
<td>469</td>
<td>46%</td>
<td>547</td>
</tr>
<tr>
<td>Topology Mapping</td>
<td>H8</td>
<td>2.3,2.4,2.6</td>
<td>2198</td>
<td>384</td>
<td>15%</td>
<td>733</td>
</tr>
<tr>
<td>Comparison</td>
<td>H9</td>
<td>1.8,2.3,2.6</td>
<td>4634</td>
<td>625</td>
<td>12%</td>
<td>1545</td>
</tr>
</tbody>
</table>

Table 6.22: Time taken by hypothesis

In this table, the third column lists all questions which tested a particular hypothesis and the fourth lists the total time (across all students) spent on the exercise. The fifth is the total time spent coaching and the sixth is the percentage of the time spent on the questions concerning a hypothesis which was spent in coaching.

It is immediately obvious that H1, H2, H5, H6, H7 and to a lesser extent H8 were completed relatively quickly. The exercises which took longest were those associated with H3 (“Behind”), H4 (“Activity-type-progress”) and H9 (“Comparison”). H7, the Selectivity question tied to question 1.10, required by far the largest relative amount of coaching. However, the total amount of coaching required was not much less than that required for H3 and H4, and actually less than that for the most conceptually novel topic, namely H9 (“Comparison”).

It is probably fair to say that 1.10 required a great deal of coaching because it invited open comment on a fairly abstract topic, namely the structure of the learner model, rather than any difficulties with H7.

6.6 Discussion concerning questions with no tied hypothesis

Several questions were asked with no explicit ties to particular hypotheses posed by this thesis. These were included, first, to determine whether particular interface elements were actually helpful; and second, to invite participant reflection on the user interface and how it might be improved in future projects.
Question 1.2 - Guided Learning

Question 1.2 references the element which is indicated with a red label on the default visualisation. This item indicates the system’s suggestion for “what to learn next” and is explained in the system help.

All participants expressed puzzlement at this and had to be either be directed to the help to determine the meaning of the text. Some participants failed to find even the explanation in the help text and needed to either be directed explicitly to the relevant section of the text or verbally informed of its meaning.

The different border in the selected circle also caused some confusion with this item, with several participants searching for a meaning for this distinct from the red text.

We conclude that this is a feature which should be either eliminated or reworked in any future implementation. The lack of context tells the user that the item selected is important without suggesting why this is the case, and this causes more confusion than clarification, exacerbated by the dashed border of the selected learning outcome.

Question 1.6 - Direction to Expert interface

The system defaults to the “simple” interface which presents a much simpler list of options and does not permit use of reference models.

P2 commented “I don’t like this expert mode - it’s confusing” when first using the interface in Q1.7. Several students found the Expert Mode interface confusing, although initial confusion faded after initial experiences in navigation.

While certain parts of the Expert interface are critical (such as the Reference model), we conclude that addressing certain elements of this interface (such as the login dialog and the terminology surrounding Resolvers) is necessary before it sees widespread use. One possible option is to make the current dual selection between “Simple” and “Expert” into a broader list of possible applications, each with its own simplified interface.

Question 1.9 - Alternative Visualisations

Question 1.9 gave all users an opportunity to compare the default visualisation against three prototype alternatives. The alternatives presented were:

- A minor variation to the primary visualisation, with text slanted at a fifteen degree angle in an attempt to reduce text overlap.

- A tree-based visualisation using the D3 library’s “dendrogram cluster” tree visualisation as shown in Figure 5.2. This option was preferred by three of the five users in our study. P1: “Yeah, it’s easier to just sort of scroll down and see what I’m missing... I quite like the dendrogram cluster, but it doesn’t have nice animations.” P2: “You know what - I like the dendrogram, even though it’s kind of... jumbo.”
A “sunflower” layout using the D3 library’s sunflower layout. This layout was preferred in principle by one user, but due to the prototype’s minimal text labeling was not regarded as usable.

The default visualisation was only preferred by a single user. Four of the five participants saw value in the function of the circular treemap representation as an overview. This group included three of those who preferred the dendrite view for general use. (The fifth did not comment.)

Availability of different visualisations proved to be a minor aid in understanding the primary visualisation. P2 noted “It only became evident to me when I saw a different model that maybe the bubble had that” (“That” being a hierarchical structure.)

**Question 1.X - Free comment on interface**

P1: "I like the idea that it’s all there. That’s good. And I’m happy that things aren’t hidden from me."

Several participants commented on the default lack of labeling for the bubble interface. This lack of labelling was the primary reason given for the dendrogram alternate visualisation being popular.

**Question 2.X - Free comment overall**

After the basic thinkaloud was complete, the participants were additionally asked if they had any other general comments.

While this was free-form, we regard these comments as perhaps the most indicative for overall response to the interface and concepts used. The interface was designed for intuitive use, but on the assumption that its underlying concepts were already understood. Most criticisms were aimed at interface details that can be addressed relatively easily or were understood by the end of the problem set. The capabilities, on the other hand, offered drew positive comment from almost all participants.

All participants indicated that the software has an extremely steep learning curve. Four of the five also indicated that they saw genuine value in the capabilities offered. (P4 did not comment specifically on this.)

All also had individual suggestions for improvements to the User Interface, varying from changes in button and panel placement to alternative ways to represent true/false values and renaming of elements (for example, from “Main/Reference” Model to “First/Second” Model.)

Table 6.23 lists some of the comments made by particular participants.
<table>
<thead>
<tr>
<th>Participant</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>I got the hang of it I think at the end. It was interesting being able to compare students with each other. I think I’d be interested to do that as a tutor.... sometimes it’s difficult to tell whether the most outspoken student is the smartest student.</td>
</tr>
<tr>
<td>P2</td>
<td>I think with a few user interface tweaks it would help... maybe feedback when I changed stuff</td>
</tr>
<tr>
<td>P2</td>
<td>With the comparing one, it would be nice if I could have two side by side.</td>
</tr>
<tr>
<td>P2</td>
<td>I thought it was interesting overall. I think if a teacher or a tutor was teaching a course, and they ... if they were using the dendrogram, and they were sitting with the student, I think it would help quite a lot. ... It might be a hard sell though, because it requires time.</td>
</tr>
<tr>
<td>P2</td>
<td>Actually I wouldn’t use the bubble model. I don’t particularly like the bubble.</td>
</tr>
<tr>
<td>P3</td>
<td>I see this [as] very useful for the tutors and also the students by themselves as well to... appreciate how much achievement they have and for know about their service, their status I mean for some specific tasks. And also it having some overview of their achievement in here so they can easily know about their achievement level as well.</td>
</tr>
</tbody>
</table>
| P3          | P3: It looks like too much information, so I have to read and I have to understand hard, but after once I use this one and then I can understand and I can realise about that, so it’s not a big problem I think? But the first time it’s very, try to learn about and try to understand about the terms as well. 
Interviewer: So basically, what you’re saying is it has a very steep learning curve. 
P3: Yeah, yeah, yeah, yeah, yeah. For the first time. And then next time, it doesn’t matter. |
| P4          | It takes some time in training to understand the interface. The rest is fine. |
| P5          | In terms of the colours, like the description for the colours... maybe it’s just me, but it took me a while to think in terms - maybe... I dunno. |
| P5          | I dunno, I feel like numbers... it needs some numbers... that’s the only bit unhappy about it. It’s like, there’s no numbers to tell me, Bob is like, 60% of the con. And maybe default colours I guess? |
| P5          | But no, I like bubble to be honest. |
| P5          | But other that that, yeah, it’s cool... very cool |
I think if I’m a tutor, um, I MIGHT use it to compare it with someone else, but... I would more likely to be comparing a person with the rest of the class. So if there’s a “rest of the class” [model] that would be great.

I think it’s definitely useful for the tutor to have that [access] but as a student... if I’m a very good student then I would totally use it, but if I’m a last-minute Larry...? then I would use it just before the exam. But then, it’s a very good use because I can see that... I need to study for Operating Systems. But it’s useful. But that suggestion of where to go next - I feel like you have to put that somewhere else.

Table 6.23: Free comments

6.7 Conclusions regarding Individual Hypotheses

H1: Overview

This was initially tested in 1.1 (for alice) and 2.1 (for bob), with 1.10 seeking free-form comments and 2.4 seeking interpretation of the overview in the context of a comparison (H8).

Participants had no trouble interpreting the overall progress of participants from the default circular treemap visualisation. There was some minor confusion for P4 as to whether “known” materials were represented by black dots or white dots, but this confusion was cleared without explicit prompting within a few seconds.

We judge the system’s success with H1 to be CONFIRMED.

H2/H3: On-track and behind

These two hypotheses are essentially inverses, but we list H3 separately for some questions where there is an obvious lack of progress (2.2, 2.3). Progress of the student model is also examined in 1.3 and 2.2.

For 1.3, which was intended to direct participants to the Search function, P3 zoomed in to the relevant part of the model where other participants used the component search function. However, since the word "for" was found in many component descriptions, P2 and P4 used a tactic of hovering over each item in the model which was quite slow.

2.2 concerns the “bob” model, which appears to be empty with the default visualisation parameters. Only one participant found the correct “Resolver” setting without heavy hinting or explicit direction. However, once this was chosen 2.3 was understood without significant difficulty (although P2 had some minor issues.).

We judge H2/H3 to be CONFIRMED WITH RESERVATIONS< due to the difficulty encountered by some participants in locating the relevant interface.
functions.

**H4: Activity-Type-Progress**

H4, testing contextualised progress, was tested in questions 1.7, 2.3, 2.4 and 2.6. H4 essentially concerns progress within a fixed subtopic; its solution may be approached either using a filter or suitable choice of a Curriculum Reference Model.

1.7 introduced use of a Reference Model and that question and those following explored use of the colour scheme. Clear understanding of use of a reference model was clearly understood in the 2.x series, particularly in 2.6. Unfortunately, 2.6 involves use of both a Reference Model and a Filter, and all of the participants, in one form or another, initially approached the problem solely using a Reference Model. Unfortunately this was incorrect as simultaneous use of a User Reference Model and a Curriculum Reference Model is not possible.

This was a clear indication that Reference Models were clearly understood by the end of the problem set but Filters were not. Any future tutorial needs to emphasise both and their relative advantages.

We judge H4 to be CONFIRMED WITH RESERVATIONS - primarily in terms of poor understanding of filters and their use.

**H5: Act**

H6, the location of learning resources, was tested only by question 1.5. As the previous question (1.4) explored the Evidence dialogue box which contains the direction for learning resources, this did not present a problem for most participants. P5 failed to note the link initially and needed mild direction.

We judge H5 to be CONFIRMED.

**H6: Openness**

H6 concerns the ability to interact with the learner model, including making changes to it. It was tested primarily by question 1.4, requiring use of the evidence dialogue box.

This was handled with little trouble for all participants except P5, who had some initial trouble locating the topic and did not initially examine the evidence dialogue in detail.

We judge H6 to be CONFIRMED.

**H7: Selectivity**

H7 relates to selecting a small part of the model to be viewed using the filter function, tested in question 1.7.
P1 and P2 were both confused by variant wording between the interface and the problem set, where initially the problem set initially asked for “self-assessment exercises” and the interface used “MOOC exercises.” Once this wording was adjusted to match, later participants found the filter much more easily.

H7 is essentially a special application of H8.

**H8: Topology Mapping**

H8 assumes use of a filter (2.3, 2.4, 2.6) but under many circumstances the problem can also be approached via an equivalent Reference Model. We found that Reference Models were in practice used preferentially, particularly in 2.6, which requires use of both. Essentially all participants made this mistake.

Once the mistake was corrected participants had no trouble interpreting the results. Essentially this was a training issue (in that the interface really requires training, but was actually approached without prior exposure or explicit instruction.)

We judge this H7/H8 to be CONFIRMED WITH RESERVATIONS, as the interface requires adjustments.

**H9: Comparison**

H9 was explicitly examined in questions 1.8, 2.3, 2.5 and 2.6, where 2.5 and 2.6 compare against a User Reference Model (alice vs. bob) and the others compare against a Curriculum Reference Model.

There was some initial confusion about use of colours, although the detailed analysis asked of participants in 1.8 seems to have made the later use of the Reference Model function much easier for the participants. The default colours drew some criticism; many preferred use of green as the primary “true” colour

Participants’ approaches to 2.6 favoured use of Reference Models over Filters. As such the problem set may have been unbalanced in its use of these elements; any future study needs to emphasis use of both functions, particularly as the filter function does not require a model load and as such is much faster than use of a reference model in the current implementation.

We judge H8 to be CONFIRMED.

**H10: Curriculum**

H10, comparison of curricula, was not tested by this study.

6.8 Conclusions from the Study

While this study is small, its participants all believed it to contain useful capabilities in the abilities provided, particularly in the ability to directly compare learner models.
The “bubble” representation was not particularly popular, although several participants observed that the favoured alternative, the dendrogram tree, would be unwieldy with larger models. This strong preference for an alternative visualisation suggests weaknesses in the current circular treemap interface. However, it also supports the integration of multiple visualisations and layouts into the learner model viewer.

Several criticisms were made of the interface where interface elements were not used as expected or contained insufficient intrinsic guidance.

All participants were able to navigate the interface and perform the described tasks, with levels of prompting required ranging from fairly minor (P1, who was also by far the fastest to complete the study) to quite significant (such as P5). Most participants indicated that while the system was initially difficult to grasp, once understood it was much easier to navigate.

The study demonstrates that the visualisation presented by MOOClm fulfils its hypothesised objectives and that the key innovations of the MOOClm platform, primarily the comparison function, make a useful contribution.
Chapter 7

Conclusion

We have here described our design and implementation of an OLM designed for use with MOOCs as part of a lifelong learning process. We have demonstrated that our OLM allows us to answer our key questions that may be asked by a student or educator and we have implemented links between this OLM and MOOC with some degree of success.

In Chapter 2 we explored briefly the literature surrounding the use of MOOCs, of Open Learner Models, and some of the visualisations which can and have been adapted for representing an OLM.

In Chapter 3 we then proposed a conceptual model for integration of MOOCs with an OLM, with a view towards lifelong use. Chapter 4 expanded on this with a detailed design showing how the component parts of such a system would interact. We describe key components of our design and how they may be used to extend the usefulness of very large Open Learner Models.

This design includes three key ideas with limited exposure in the literature. These are use of a circular treemap visualisation for an OLM, selection of topological filters to explicitly restrict the part of an OLM being viewed, and use of a second or Reference Model for comparison of OLMs.

Use of the circular treemap visualisation as a basis for an OLM interface appears to be new; we believe that it has advantages in the handling of very large learner models that have received very little attention.

While the circular treemap visualisation was not well received in our user study, the main criticisms raised were of use as a detail view rather than the overview functions to which it is better suited. An interface which permits use of multiple visualisations allows the learner to switch freely between them. The learner is then free to optimise their interaction with the OLM by selecting an appropriate interface.

The increase in model size in turn leads to our second novel element: the use of filters, or topological maps, to selectively present particular portions of the OLM to the student. While filtering of larger models into smaller, more manageable chunks is certainly not new, this has primarily been done on a heuristic basis to present certain “interesting” parts of a learner model, typically as a
means of directing learning. While directing learning is certainly a critical goal, in doing this we risk losing some of the openness that makes OLMs valuable. An OLM is not truly open if parts of the model are hidden from the student.

While our range of filters is relatively small, the ability of the circular treemap representation to “zoom in” to parts of a model may substitute for a wider variety in the short term.

The third novel element is, again, not entirely novel, but a development of existing ideas. Comparison of Learner Models between students has been in the literature for over a decade, but generalising the concept into the idea of a “Reference Model” appears to be a novel contribution. A Reference Model can be built not only on the basis of an individual or notional student, but also on the basis of curriculum or assessment components.

Chapter 5 then describes how we implemented this design within a framework integrating the ACM CS2013 standardised curriculum as a baseline, EdX as a MOOC platform and Personis as an OLM backend.

In Chapter 6 we then describe a small thinkaloud study. We show that the core capabilities of our implementation were understood and valued despite several interface flaws that were identified by the study. The thinkaloud study serves as an independent check on how well this design succeeds in meeting our core goals.

Future Work

The user study highlighted certain components of the interface design which need to be addressed. A basic tutorial, or perhaps a good labelled diagram of the interface, may assist with the system’s steep learning curve. Certain functions such as the nature of the login interface and choice of resolvers require significant clarification. The highlighting used for “next thing to learn” needs to either be eliminated or broken out into an explicit selection, possibly cycling through the “next several things to learn.”

Building certain types of Reference Model is problematical. If we seem to create a Reference Model representing overall class progress, on what basis should it be built? If median class performance, how do we handle this in the context of the high dropout rate of typical MOOCs?

This does suggest an intriguing possibility, which we might call the Population Learner Model: a “learner” model integrating the learning data for a student population, with population-level equivalents to Personis’ Resolvers which aggregate student data for comparison. While this sort of analysis is routine, representing such an assembly in relation to a learner model and its model curriculum introduces a novel viewpoint for examining such data.

The process of mapping between learning objectives and course elements is currently quite tedious. This is very simple in principle, and in some ways has already been done in existing curriculum mapping software such as ProGoSS; the problem lies making it easy for the MOOC designer to create this mapping without undue difficulty. We see two obvious paths: export course
elements from the MOOC to be used in curriculum design software, or import learning objectives into the MOOC and modify the MOOC platform to permit assignment of learning objectives.

We have yet to demonstrate actual integration of multiple MOOCs using a learner model. One of the main requirements for a system such as that described here is to actually demonstrate interoperability with multiple MOOCs. While we have attempted to design towards such interoperability, it remains to be seen whether this planning was effective and useful.

We also believe that the visualisation space for OLMs still contains space for new OLM interfaces. Our basic design makes expansion into new interfaces relatively straightforward. The “icicle” interface has some promise here, and it is possible that some of the visualisations dismissed as overly crowded in our background chapter may be rendered useful with careful modification.

We believe that the ability to switch between user interfaces “on the fly” is advantageous to the student. Selection of an optimal set of interfaces to be made available for an OLM would appear to be an interesting area of research.

Summary

Our contribution demonstrates the creation of an Open Learner Model suitable for use with MOOCs in a lifelong learner context. The MOOClm framework then integrates this OLM with a MOOC. Key innovations of this framework are the use of filters to limit visible scope, and the use of a “Reference Model” to explicitly compare learner models.
Bibliography


144


G Siemens and S Downes. Connectivism and connective knowledge online (course: Cck08), 2008.


Julia Wilkowski, Amit Deutsch, and Daniel M Russell. Student skill and goal achievement in the mapping with google MOOC. In *Learning@Scale 2014*, 2014.


Appendices
Appendix A

**Nqv (New QuickView)**

nqv was an earlier visualisation design which inspired some of the later design choices made in MOOClm.

nqv (NewQuickView), was an initial attempt to expand and refine the basic design of the original qv [Cook and Kay, 1994] but with a number of tweaks intended to simplify navigation (A.1). It is included here as parts of the design inspired later design choice - as frequently by its failures as by its sparse successes.

![User Model for Student](image)

**Figure A.1: nqv**

- Each model component or sub-component is linked to its parent by a grey-shaded link which indicates how much of the model beneath that point is
populated or “true”. This enables finding those areas of the model which are wholly or partially populated, or completely blank.

- Vertical size of each parent node is scaled non-linearly to hint at the size of the underlying sub-model. This gives the learner a visual clue as to the amount of material lying beneath any given link.

- The viewer provides a dropdown selector which can be used to choose from any of several arbitrary mappings for the model. This idea was carried forward into the final MOOClm design.

- As with the original qv, each subtree can be collapsed or expanded to provide a narrow or broad view of the model.

While these characteristics were intended to provide visual hints as to particular parts of the model, the scalability issue remained largely unresolved and comparisons against other learner models were not supported.
## Appendix B

### Exam Learning Objectives

<table>
<thead>
<tr>
<th>Q No.</th>
<th>Max Mark</th>
<th>Topic</th>
<th>Coded Learning Outcome (under Education/ComputerScience)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>man command</td>
<td>UNIX/BourneShell/Core/Manual</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Pointers</td>
<td>C/Types/Pointers</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Memory/storage management</td>
<td>C/MemoryManagement/StaticAndConstant</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>malloc/free (heap memory manage-</td>
<td>UNIX/SystemsProgramming/ MemoryManagement</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>C unions</td>
<td>C/Types/Unions</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Shell pipes &amp; redirection</td>
<td>UNIX/BourneShell/ShellProgramming/ PipesAndFDArithmetic</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>C strings</td>
<td>C/Types/Strings</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>C strings</td>
<td>C/LibraryFunctions/Printf</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>Stack frame organisation</td>
<td>C/MemoryManagement/ StackHowSpacesAssigned</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>Memory/storage management</td>
<td>C/MemoryManagement/StackAndHeap</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>C for loops</td>
<td>C/Commands/Control</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>C structures &amp; memory layout</td>
<td>C/Types/Struct</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>Version/source control systems</td>
<td>SourceCodeControl/Mercurial/ TaggedChangesets</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>Build utilities /make</td>
<td>C/CompilingMultipleFiles/MakeRules</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Q No.</th>
<th>Max Mark</th>
<th>Topic</th>
<th>Coded Learning Outcome (under Education/ComputerScience)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
<td>C declarations &amp; storage</td>
<td>C/MemoryManagement/StackAndHeap</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>C strings &amp; memory management</td>
<td>UNIX/SystemsProgramming/MemoryManagement</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>C for loops &amp; array organisation</td>
<td>C/Commands/Control</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>C for loops &amp; array organisation</td>
<td>C/Types/Array</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>C for loops &amp; array organisation</td>
<td>C/Operators/ArrayReference</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>Memory/storage management (&amp; const strings)</td>
<td>C/MemoryManagement/StaticAndConstant</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>C strings &amp; memory management</td>
<td>C/Types/Strings</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>C strings &amp; memory management</td>
<td>UNIX/SystemsProgramming/MemoryManagement</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>Threads &amp; memory allocation</td>
<td>Parallelism/Tier1/04</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>Code tracing</td>
<td>C/Operators/Assignment</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>Process/thread deadlocks</td>
<td>CommunicationandCoordination/Tier2/03</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>Resource monitors</td>
<td>(Not covered in MOOC)</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>Process/thread deadlocks (self-deadlocks)</td>
<td>CommunicationandCoordination/Tier2/03</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>Barriers</td>
<td>(Not covered in MOOC)</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>Atomic operations</td>
<td>ParallelArchitecture/Electives/06</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>Counting semaphores</td>
<td>Concurrency/Tier2/05</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>Condition variables</td>
<td>(Not covered in MOOC)</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>Load balancing</td>
<td>ParallelPerformance/Electives/01</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>Performance bottlenecks</td>
<td>(Not covered in MOOC)</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>Locking hierarchies</td>
<td>CommunicationandCoordination/Electives/12</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>Monitors and semaphores</td>
<td>CommunicationandCoordination/Electives/12</td>
</tr>
<tr>
<td>33</td>
<td>16</td>
<td>C functions, memory management, loops</td>
<td>C/Types/Strings</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Q No.</th>
<th>Max Mark</th>
<th>Topic</th>
<th>Coded Learning Outcome (under Education/ComputerScience)</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>8</td>
<td>Code tracing, malloc/free, pointers, arrays</td>
<td>UNIX/SystemsProgramming/ MemoryManagement</td>
</tr>
<tr>
<td>34</td>
<td>8</td>
<td>Code tracing, malloc/free, pointers, arrays</td>
<td>C/Types/Pointers</td>
</tr>
<tr>
<td>35</td>
<td>8</td>
<td>UNIX shell scripting</td>
<td>UNIX/BourneShell/ShellProgramming/ PipesAndFDArithmetic</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>Memory management, C structures, arrays</td>
<td>C/Commands/ FunctionsAndFunctionInvocation</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>Memory management, C structures, arrays</td>
<td>C/Types/Struct</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>Memory management, C structures, arrays</td>
<td>C/Types/Pointers</td>
</tr>
<tr>
<td>37</td>
<td>12</td>
<td>Process/resource locking &amp; threaded execution</td>
<td>CommunicationandCoordination/Electives/12</td>
</tr>
<tr>
<td>37</td>
<td>12</td>
<td>Process/resource locking &amp; threaded execution</td>
<td>CommunicationandCoordination/Tier2/09</td>
</tr>
</tbody>
</table>
Appendix C

Characteristics of Full Learner Model

This appendix describes the number or learner objectives at each “level”, or depth, of the tree structure that describes the topology of our OLM.

Note that the breakdown here refers to three different logical topologies for our OLM.

1. **Full**: The full OLM, including all ACM and supplemental learning objectives.
2. **MOOC**: The OLM as restricted only to those items covered in the MOOC.
3. **COMP2129**: The OLM as restricted to those items covered in the MOOC or the final exam.

The full model includes two high-level structural components (“Education” then “Computer Science”) which are omitted for the restricted models, in order to make better use of the working space.

<table>
<thead>
<tr>
<th>Depth</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural Links (Full)</td>
<td>1</td>
<td>18</td>
<td>166</td>
<td>224</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Structural Links (MOOC)</td>
<td>1</td>
<td>4</td>
<td>11</td>
<td>25</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Structural Links (COMP2129)</td>
<td>1</td>
<td>6</td>
<td>16</td>
<td>31</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Learning Objectives (Full)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1220</td>
<td>55</td>
</tr>
<tr>
<td>Learning Objectives (MOOC)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>113</td>
<td>48</td>
</tr>
<tr>
<td>Learning Objectives (COMP2129)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>108</td>
<td>48</td>
</tr>
<tr>
<td>Learning Objectives in Exam</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>3</td>
</tr>
<tr>
<td>Learning Objectives in Self-Test</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>10</td>
</tr>
<tr>
<td>Learning Objectives in Lectures</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>108</td>
<td>48</td>
</tr>
<tr>
<td>Learning Objectives in Exam+Self-Test</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>38</td>
<td>11</td>
</tr>
<tr>
<td>Learning Objectives in Exam+Lecture</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>117</td>
<td>48</td>
</tr>
</tbody>
</table>
Appendix D

Restricted MOOC Ontology (Phase 1)

Items marked with a “*” are “parent nodes” which do not have their own values, but structure the ontology.

This is the “limited ontology” which includes all learner outcomes included in the MOOC or the 2014 final exam

1. **OperatingSystems**: Operating Systems (OS)
   1.1. **UNIX**: Knowledge about UNIX and UNIX-like Operating Systems
      1.1.1. **BourneShell**: UNIX command line usage
      1.1.1.1. **Core**: UNIX command line usage - core commands
      1.1.1.1.1. **BasicCommands**: Can demonstrate entry level UNIX commands: cd, ls, cat, cp, rm, mv, echo
      1.1.1.1.2. **FilePermissions**: Demonstrate ability to read UNIX file permissions and modify them with chmod
      1.1.1.1.3. **Globbing**: Can demonstrate use of shell globbing - Using wildcards such as * and []
      1.1.1.1.4. **GrepCommand**: Can correctly use grep for searching files for regular expressions; flags such as “-i”
      1.1.1.1.4.1. **Manual**: Knows how to access the online manual: the man command
      1.1.1.2. **CoreTools**: UNIX command line usage - Fundamental tools
      1.1.1.2.1. **DiffAndSDiff**: Demonstrates knowledge of use of diff & sdiff
      1.1.1.2.2. **EchoAndPrintf**: Can successfully use the echo and printf commands
      1.1.1.2.3. **FindCommand**: Understands use of the find command to recurse through all directories; understands use of locate & du
      1.1.1.2.3.1. **IntermediateCommands**: Demonstrates knowledge of next-to-entry level UNIX commands: head, tail, diff, chmod, sort, fmt, wc
1.1.2.4. **Other Useful Commands**: Demonstrates knowledge of other key useful commands - sort, cut, tr, comm

1.1.2.5. **Top Command**: Can demonstrate use of the `top` command to monitor system processes & status

1.1.3. **Shell Programming**: UNIX command line usage - Shell Programming

1.1.3.1. **Backquotes**: Understands use of Backquotes for command substitution

1.1.3.2. **Control Structures**: Can successfully use UNIX shell Control structures - if, for, case, while

1.1.3.3. **DotSlashPath**: Understands use of ./filename to work around accidental strange file names

1.1.3.4. **Expr Command**: Understands use of expr for arithmetic in shell - eg. i=`expr $i - 1`

1.1.3.5. **Here Documents**: Knows how to use Here documents - command

1.1.3.6. **PATH Variable**: Demonstrates knowledge of the PATH shell variable controlling where the shell looks for commands?

1.1.3.7. **PS1**: Demonstrates knowledge of how to change the shell prompt by setting PS1

1.1.3.8. **Parentheses Subshell**: Use of parentheses () for creating a subshell

1.1.3.9. **Pipes and F D Arithmetic**: Understands use of pipes (|) and numbered I/O redirection (1>&2)

1.1.3.10. **Read Command**: Knows how to use the read command to read one line from stdin

1.1.3.11. **Redirections and Pipes**: Understands how to write a UNIX command so that it: reads from a file (<), writes to and truncates a file (>), appends to a file (>), pipes its output to another program (|)

1.1.3.12. **Shell Arguments**: Demonstrates knowledge of how to use Argument variables in shell scripts - $1, $2 etc, $*, $@

1.1.3.12.1. **Shell as Interpreter**: Understands the role of the shell as a UNIX command interpreter

1.1.3.13. **Shell Variables**: Understands use of Shell variables; shell environment, standard shell variables PATH, HOME & others; VAR=value / $VAR-NAME

1.1.3.14. **Test Command**: Knows how to use the test command; alias; don’t call your own program test!

1.1.3.15. **Test Command Flags**: Knows flags for test command in testing file/dir properties

1.2. **Fundamentals**: Fundamental information about UNIX

1.2.1. **Core Components**: Can describe the core components of UNIX (kernel, shell, file system...)

1.2.2. **Directory Root**: Can explain that UNIX filesystem is rooted at / with path components separated by /

1.2.3. **File Types**: Can describe the main types of file under UNIX

1.2.4. **Hash Bang**: Can explain use of #!/shell/path at beginning of shell files
1.1.2.5. **HomeDirectory**: Understands that the (tilde) character can be used as shorthand for your home directory, as can $HOME

1.1.2.6. **ShellStartRead**: Demonstrated knowledge that of which files the shell reads when starting up

1.1.3. **Origin**: Origins of UNIX

1.1.3.1. **AustralianContributions**: Can outline the core contributions made by Australian researchers in the design and authoring of UNIX

1.1.3.2. **DevelopedBy**: Can outline who authored UNIX (primarily Kernighan and Pike)

1.1.3.3. **LinuxKernel**: Can state the name of the main central controlling program in Linux (the kernel)

1.1.3.4. **OSDerivatives**: Can name two modern operating systems based, directly or indirectly, on UNIX (such as Linux and iOS/Darwin)

1.1.4. **SystemsProgramming**: Basics of UNIX Systems Programming

1.1.4.1. **ExecCalls**: Can describe the `exec` family of system calls: `execl`, `execle`, `execv`, `execve` - to execute a separate program

1.1.4.2. **FileDescriptors**: Can describe use of the basic file descriptors - 0 (stdin), 1 (stdout) & 2 (stderr) as well as how others are created, closed and used

1.1.4.3. **Fork**: Can correctly use the `fork()` system call and understands the differences in its behaviour between parent and child

1.1.4.4. **IODeviceFunctions**: Can correctly use UNIX I/O system functions - `creat`, `open`, `close`, `read`, `write`, `ioctl`

1.1.4.5. **MainAndEnvp**: Can describe how the `envp` array is passed to `main()`

1.1.4.6. **MemoryManagement**: Understands basic UNIX (heap-based) Memory management using `malloc`, `calloc`, `free`, `realloc`

1.1.4.7. **Perror**: Can correctly use `perror` for error reporting when a system call returns an error

1.1.4.8. **PipeBuffering**: Understands how data is usually buffered when passing through a pipe

1.1.4.9. **PipeSystemCall**: Can correctly use the pipe system call; read on filedes 0, write on filedes 1

1.1.4.10. **PipesAndSignals**: Understands use of pipes for active communication between processes, vs. use of signals for interrupts

1.1.4.11. **PipeSystemCall**: Can correctly use pipes for IPC

1.1.4.12. **SignalFunction**: Can correctly use the `signal()` system call for catching interrupts

1.1.4.13. **SignalsAndKill**: Understands use of signals & interrupts - `kill` command & `syscall`

1.1.4.14. **UncatchableInterrupts**: Understands that some signals cannot be caught, notably SIGKILL (kill -9)

1.1.4.15. **WaitAndWaitpid**: Can correctly use `wait()` & `waitpid()` to wait for a child process to exit

2. **ProgrammingLanguages**: Programming Languages (PL)

2.1. **C**: Knowledge about the C Programming Language

2.1.1. **Commands**: C control commands
2.1.1. **Control**: Knows of the C control flow structures: while, for, do... while, switch, if...else, break/continue

2.1.1.2. **DeclarationLocation**: Understands that in C, variable and other declarations only occur at the start of program block delimitated using parentheses

2.1.1.3. **ForLoop**: Knows the details of the C-style for loop - for ( initialise; condition; postoperation)

2.1.1.4. **FunctionsAndFunctionInvocation**: Can use and declare C functions & knows how functions are invoked

2.1.1.5. **VariableScope**: Knows how C handles Variable scope - local, global and extern declarations

2.1.2. **CommonErrors**: Common errors made while programming in C

2.1.2.1. **AllocatingWrongAmountOfMemory**: Knows of the common error of specifying incorrect memory allocation sizes - usually allowing for extra character in char* for terminating None

2.1.2.2. **DereferencingFreedPointer**: Can explain the Common error of Dereferencing a pointer after it has been freed

2.1.2.3. **EqualAsComparison**: Can explain the Common error of Using = (the assignment operator) for comparison

2.1.2.4. **FailingToCheckMallocSuccess**: Can explain the Common error of failing to check for failure on memory allocation

2.1.2.5. **FailureToParenthesiseMacros**: Can explain the Common error of Failure to parenthesise #define macros

2.1.2.6. **Fencepost**: Can explain the Common error of Fencepost errors, where a fixed amount of memory is under-allocated due to failure to allow for overflow conditions such as the terminating None in a string

2.1.2.7. **FreeUnallocatedMemory**: Can explain common errors in Memory management such as freeing up unallocated memory

2.1.3. **CompilingMultipleFiles**: Compiling multiple files in C with make and the linker

2.1.3.1. **Ar**: Can Use the UNIX ar command to create static libraries

2.1.3.2. **CommandLineForGCC**: Can describe the important command line arguments for gcc, particularly -o and -c

2.1.3.3. **CompilingPipeline**: Can explain the structure of the compiling pipeline: preprocessor, compiler, assembler, linker

2.1.3.4. **LinkerAndObjectFiles**: Can compile multiple files together using the linker, .o files, extern

2.1.3.5. **MakeDefaultRules**: Can explain the use of default Make rules: (.c -> .o and so on)

2.1.3.6. **MakeRules**: Knows how rules are structured in a Make rules: target, dependencies, action

2.1.3.7. **MakeRulesMultipleTargets**: Can correctly use multiple targets in a single rule when using Make

2.1.3.8. **MakeRulesNoDependents**: Knows how to use targets with no dependencies in Make, e.g. “clean”
2.1.3.8.1. MakeVariables: Can correctly use Make variables (VARIABLE = value) and predefined command-line variables

2.1.4. LibraryFunctions*: Common C Library Functions

2.1.4.1. NULLAndVoidStar: Knows of how The NULL pointer and void * are used in C

2.1.4.2. PassByValue: Can explain that in C, all functions are pass by value and pointers must be used for pass by reference

2.1.4.3. PointersToFunctions: Can effectively use Pointers to functions

2.1.4.4. PointersToPointers: Can correctly use pointers to pointers & other multiple indirection

2.1.4.5. Printf: Can correctly use the printf function, including % formatting & field specifications

2.1.4.6. RegularExpressions: Can build & use Regular expressions - designed to match text against a pattern; powerful but difficult to use; knows of the regcmp and regex library functions

2.1.4.7. Search: Can demonstrate use of the standard function bsearch - binary search

2.1.4.8. Sorting: Can use the qsort for quick sorting operations in C; knows how to supply a comparison function

2.1.4.9. SprintfAndScanf: Can correctly use sprintf, scanf, sscanf for flexible I/O & basic parsing

2.1.4.10. StringFunctions: Can differentiate and use the standard string functions - strcpy, strncpy, strcat, strncat, strcmp, strncmp, strstr

2.1.4.11. VarArgs: Knows how to use the Varargs library in C to create C functions with a variable number of arguments

2.1.4.12. ZeroFalseNonzerotrue: Can describe C’s use of Use of 0 and nonzero as false/true; if (0) ...idiom

2.1.5. MemoryManagement*: Memory Management in C (Stack, heap & static)

2.1.5.1. StackAndHeap: Can explain how memory is allocated across the stack, the heap, how this is done for function calls, local variables and static variables

2.1.5.2. StackHowSpaceIsAssigned: Can explain how space is assigned on the stack during function operations

2.1.5.3. StaticAndConstant: Can explain the difference in handling and memory allocation off variables assigned space on program initialisation

2.1.5.4. WrapperFunctionsForMemorySafeguards: Can effectively use wrapper functions to incorporate checks for memory management

2.1.6. Operators*: C Operators

2.1.6.1. ArrayReference: Can use the C Array reference operator: []

2.1.6.2. Assignment: Can correctly use C Assignment operators and assignment/operation combined operators: = += -= *= /= %= &= |= ^= *=« ^= »=

2.1.6.3. Bitwise: Can correctly use the C Bitwise operators: & | ^ « »

2.1.6.4. CommaAndTernary: Can correctly use the C comma and conditional evaluation operators: , ?:
2.1.6.5. **Comparison**: Can correctly use the C Comparison operators: == != < <= > >=

2.1.6.6. **PointerAndRecordAccess**: Can correctly use C operators for Pointer & record access: . -> * unary &

2.1.6.7. **PrePostIncDecrement**: Can correctly use C Pre- and post- increment and decrement operators: ++ –

2.1.6.8. **Sizeof**: Can describe correct use of the sizeof operator and pointer arithmetic

2.1.7. **OriginsAndUsage**: Origins of C and basic usage

2.1.7.1. **DifferencesFromJava**: Can correctly outline the major differences between Java and C

2.1.7.2. **HowToCompile**: Can correctly compile a C program under UNIX

2.1.7.3. **MainUsages**: Can outline the class of applications for which C is primarily used for development?

2.1.7.4. **MajorApplicationExamples**: Can Name several major applications written in C such as Apache, python, the Linux kernel

2.1.7.5. **WhoAndWhereDeveloped**: Can describe Who developed the C language and where was it developed

2.1.8. **PreProcessor**: Using the C Preprocessor

2.1.8.1. **HashIncludeAndMacros**: Can correctly use the #include and #define (macro) preprocessor directives

2.1.8.2. **HeaderFiles**: Can correctly use header files in multiple-file C programs to facilitate sharing of data and definitions

2.1.8.3. **MacrosWithArguments**: Can correctly use preprocessor Macros with parameters and is aware of risks of multiple evaluation

2.1.8.4. **PredefinedSymbols**: Can correctly use key Predefined preprocessor symbols - __LINE__ and __FILE__

2.1.8.5. **PreprocessorConditionals**: Can correctly command the C preprocessor for Conditional inclusion & macros - #ifdef, #ifndef, #if, #else, #elif, #undef

2.1.8.6. **PreprocessorWithOtherLanguages**: Knows how the C preprocessor may be used for processing formats other than C

2.1.8.7. **QuotesVsAngleBrackets**: Can describe the difference between how double-quotes "" vs. angle brackets <> are handled in C #include

2.1.9. **Structure**: Structure of a C program

2.1.9.1. **Blocks**: Can explain that a C code block is enclosed by parentheses

2.1.9.2. **Comments**: Can correctly use C comments /* ... */

2.1.9.3. **MainFunction**: Can correctly state that the main() function is the primary entry point for a C program

2.1.9.4. **Semicolon**: Correctly uses the Semicolon as a compulsory statement separator in C

2.1.9.5. **StdioHeader**: Can describe which external header file includes the standard C I/O functions and interfaces <stdio.h>

2.1.9.6. **Void**: Can explain what a function returning void or with a void parameter list means

2.1.10. **Types**: Basic Types in C
2.1.10.1. **Array**: Can correctly use C arrays - declaration & access via [ ]; understands that indexing starts at 0
2.1.10.2. **Bitfields**: Can use C bitfields in structs
2.1.10.3. **Enum**: Can use enum(eration) variables in a C program
2.1.10.4. **Pointers**: Can correctly use pointers for any of the other types, including structures
2.1.10.5. **Strings**: Can correctly use C strings (as None-terminated arrays of char *)
2.1.10.6. **Struct**: Can correctly create & access C records (struct)
2.1.10.7. **Typedef**: Can describe and correctly use typedef
2.1.10.8. **Unions**: Can describe why unions are useful and knows how to declare and use them; understands the need for field in struct to indicate union type

2.2. **CodeGeneration**: PL/Code Generation
2.2.1. **Electives**: Elective Learning Outcomes in PL/Code Generation
2.2.1.1. **01**: Identify all essential steps for automatically converting source code into assembly or other low-level languages.

2.3. **FunctionalProgramming**: PL/Functional Programming
2.3.1. **Tier2**: Tier 2 Learning Outcomes in PL/Functional Programming
2.3.1.1. **04**: Correctly reason about variables and lexical scope in a program using function closures.

2.4. **LanguagePragmatics**: PL/Language Pragmatics
2.4.1. **Electives**: Elective Learning Outcomes in PL/Language Pragmatics
2.4.1.1. **05**: Discuss the need for allowing calls to external calls and system libraries and the consequences for language implementation.

2.5. **LanguageTranslationandExecution**: PL/Language Translation and Execution
2.5.1. **Tier2**: Tier 2 Learning Outcomes in PL/Language Translation and Execution
2.5.1.1. **05**: Identify and fix memory leaks and dangling-pointer dereferences.

3. **SoftwareDevelopmentFundamentals**: Software Development Fundamentals (SDF)
3.1. **DataStructures**: Knowledge about different data structures
3.1.1. **LinkedLists**: Linked Lists - records linearly linked by pointers
3.1.1.1. **InsertionOperations**: Can explain the sequence of actions for insertion operations for linked lists
3.1.1.2. **NULLTermination**: Understands the use of NULL in termination of a linked list
3.1.1.3. **Performance**: Can demonstrate understanding of performance characteristics of linked lists, including linear worst-case access cost
3.1.1.4. **Programming**: Can demonstrate programming of a simple linked list, including insert, append, and delete item operations
3.1.1.5. **Structure**: Can explain the structure of a linked list, each node consisting of a structure with data and pointer
3.1.2. **Numeric**: Representation and issues with numeric data types
3.1.2.1. **FloatingPoint**: Representation and issues with floating point arithmetic
3.1.2.1.1. **BenignCancellation**: Can demonstrate understanding of benign cancellation in floating point calculations and methods to restructure expressions to reduce effects of order of operations

3.1.2.1.2. **CatastrophicCancellation**: Can demonstrate understanding of catastrophic cancellation in floating point calculations, where inaccuracies in representation are multiplied by order of operations

3.1.2.1.3. **Components**: Can describe the components of a floating point representation, including sign, exponent and mantissa; Can describe correspondence with scientific notation

3.1.2.1.4. **DesignConsiderations**: Can explain design considerations for floating point types and explain relative advantages and disadvantages compared with fixed point

3.1.2.1.5. **DoNotTestEquality**: Understands that testing floating point values for equality is usually unwise and why this is the case

3.1.2.1.6. **Exponent**: Understands use of excess-127 encoding for the Exponent of a floating point representation (add 127 before binary conversion)

3.1.2.1.7. **IEEE754**: Can describe the IEEE-754 Floating Point Standard

3.1.2.1.8. **Infinity**: Can explain encoding of infinity in floating point representations - mantissa all zeroes, exponent all ones, sign for +/- infinity

3.1.2.1.9. **LossOfAccuracy**: Can explain loss of accuracy due to floating point representation - 1000x0.1 may not come to 100. Large numbers lose precision in lower decimal places; rounding errors

3.1.2.1.10. **Normalisation**: Understands use of normalisation in floating point representation, with a zero exponent being used to represent zero, infinity and NaN

3.1.2.1.11. **PrecisionClasses**: Can outline the different classes of floating point representation - single [32-bit], double [64-bit], extended [80-bit] precision

3.1.2.1.12. **RealWorldFailures**: Can cite real world examples of catastrophic failure due to calculation errors

3.1.2.1.13. **SignBit**: Knows that the sign bit for floating point is 0 for positive and 1 for negative numbers

3.1.2.1.14. **SourcesOfError**: Can cite possible sources of error in floating point calculations, including but not limited to: Original data, propagation error, representational errors; remaining aware of errors & their sources, of absolute & relative error.

3.2. **DevelopmentMethods**: SDF/Development Methods

3.2.1. **SourceCodeControl**: Source Code Control Systems

3.2.1.1. **Mercurial**: The Mercurial SCCS System

3.2.1.1.1. **CommonCommands**: Demonstrates knowledge of commands for use with mercurial - hg clone etc.

3.2.1.1.2. **Merging**: Knows how to merge changes & conflicts in Mercurial

3.2.1.1.3. **SCCSType**: Knows that, of the different basic types of SCCS, Mercurial is of the clone/push/pull variety

3.2.1.1.4. **TaggedChangesets**: Understands use of tagged changesets in Mercurial

3.2.1.2. **Rationale**: Reasons for using Source Code Control
3.2.1.2.1. **CheckinCheckout**: Understands difference between the two basic types of SCCS: checkin/checkout vs. clone/push/pull systems
3.2.1.2.2. **RollBack**: Understands need for rollback to previous versions
3.2.1.3. **Software**: Types of SCCS
3.2.1.3.1. **Examples**: Demonstrates knowledge of names for some common version control systems - e.g. Subversion, Mercurial, git
3.2.2. **Tier1**: Tier 1 Learning Outcomes in SDF/Development Methods
3.2.2.1. **08**: Apply a variety of strategies to the testing and debugging of simple programs.
3.2.2.2. **10**: Construct and debug programs using the standard libraries available with a chosen programming language.
3.2.2.3. **12**: Apply consistent documentation and program style standards that contribute to the readability and maintainability of software.
3.3. **FundamentalDataStructures**: SDF/Fundamental Data Structures
3.3.1. **Tier1**: Tier 1 Learning Outcomes in SDF/Fundamental Data Structures
3.3.1.1. **03**: Write programs that use each of the following data structures: arrays, records/structs, strings, linked lists, stacks, queues, sets, and maps.
3.4. **FundamentalProgrammingConcepts**: SDF/Fundamental Programming Concepts
3.4.1. **Tier1**: Tier 1 Learning Outcomes in SDF/Fundamental Programming Concepts
3.4.1.1. **01**: Analyze and explain the behavior of simple programs involving the fundamental programming constructs variables, expressions, assignments, I/O, control constructs, functions, parameter passing, and recursion.
3.4.1.2. **02**: Identify and describe uses of primitive data types.
3.4.1.3. **07**: Choose appropriate conditional and iteration constructs for a given programming task.
4. **SoftwareEngineering**: Software Engineering (SE)
4.1. **ToolsandEnvironments**: SE/Tools and Environments
4.1.1. **Tier2**: Tier 2 Learning Outcomes in SE/Tools and Environments
4.1.1.1. **02**: Describe how version control can be used to help manage software release management.
Appendix E

Think-Aloud Study: Script

This appendix describes the script which was to be used in prompting participants during the Thinkaloud study.

Interviewer Instructions

Before beginning, hand the participant the Participant Information Statement (PIS) and the Participant Instructions. Ask them to sign the PIS.

If the participant asks for assistance with the interface, indicate the relevant options and ask them for comment on how the interface could be improved. The Participant Instructions instruct the Participant to do so.
Participant Instructions

Today you are being asked to evaluate a software system for showing information about a person’s progress in learning from a MOOC, a learner model. This system is based on a second-year computer science course concerning the C programming language, the UNIX operating system environment, and some related topics (COMP2129 at Sydney University, as taught in 2014).

This study uses a "think-aloud" protocol.

You will be asked to perform certain tasks. In performing these tasks, we ask that you describe your thoughts and feelings at each step - effectively a "stream of consciousness" - and why you are taking each action. We will show you a brief video of the protocol in action before proceeding with the interactive part of the study.

We emphasise that we are testing the interface, not your own knowledge.

Please indicate if you find any step particularly easy or difficult, or whether particular interface elements are helpful or confusing. Be as clear, honest and realistic as possible; do not sugarcoat your opinions, but if you do like a feature, please say so.

If you need assistance with any task, please ask the study supervisor and indicate any faults evident in the interface design or help system. In general the supervisor should not direct you unless you ask for assistance.

Please tell your interviewer when you are ready to view the sample video before proceeding as below.

In the first stage, you will be acting as a hypothetical student, "Alice", viewing her progress in the course. This student has recently completed the COMP2129 second-year course at the University of Sydney.

The home page of the browser on your screen has been set to the interface. The username and password for your learner model are as follows:

Username: alice Password: degoz6mM3

Please attempt the following tasks. Describe your actions and your thoughts as you attempt each action.

1. Please log in using the provided username and login. What does "Alice" know well in the COMP2129 course? Give your initial impressions.

2. Determine the meaning of the red text.

3. Does the system think "Alice" knows about "for" loops in C?

4. As "Alice", you think that the system is incorrect about her knowledge of "for" loops. Can you correct this?

5. Attempt to locate learning resources concerning this topic.

6. If you have not already done so, switch the interface from “Simple” to “Expert” mode.

7. See how well Alice has covered material in self-assessment exercises in the MOOC.

166
8. Experiment with the colour selections. Explain in your own words what you think the colours mean, using specific examples of what Alice knows vs. what was covered in self-assessment exercises.

9. Try some of the alternatives listed under Visualisations. (Note that these are prototypes with only partial functionality.) Pick one of these and compare it with the default layout.

10. Please comment on the actual learner model structure. Is it too big? Too small? Are parts confusing?

   Having explored the interface thoroughly, give your overall impressions, including any criticisms you may have.

In the second part of this study, you are a tutor for Alice and a second student, “bob”.

Username: alice Password: degoz6mM3
Username: bob Password: Rh11K9Xa4

1. Log into “bob”’s model and evaluate the usefulness of the view from the perspective of a tutor.

2. Describe “bob”’s progress on the course. Is there a way to vary the view to show a more “optimistic” interpretation of the evidence? Indicate whether you believe the more optimistic view is preferable and explain your opinion.

3. Evaluate “bob”’s progress against the learning objectives tested in the final exam. Describe what the different colours mean in your own words, and your opinion of how they are used.

4. See how much of the final exam was known by “bob” as of the end of March.

5. Compare how much had been learnt by Alice and bob by the end of term.

6. Compare how much had been learnt by both solely within the topics tested by the exam.
Appendix F

Think-Aloud Study: Ethics Approval

The following pages include the ethical approval for the study performed for this thesis and for a later modification of the study protocol.
Dear Judith

I am pleased to inform you that the University of Sydney Human Research Ethics Committee (HREC) has approved your project entitled "Building user models from MOOC data".

Details of the approval are as follows:

**Project No.**: 2015/277  
**Approval Date**: 6 June 2015  
**First Annual Report Due**: 6 June 2016  
**Authorised Personnel**: Kay Judith; Kummerfeld Robert; Cook Ronald;

**Documents Approved**:

<table>
<thead>
<tr>
<th>Date</th>
<th>Type</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/05/2015</td>
<td>Other Instruments/Tools</td>
<td>Snapshot of MOOC page containing PIS and link to viewer tool</td>
</tr>
<tr>
<td>04/05/2015</td>
<td>Other Instruments/Tools</td>
<td>Snapshot of MOOC page for explicitly revoking consent</td>
</tr>
<tr>
<td>10/04/2015</td>
<td>Participant Consent Form</td>
<td>PIS Building User Models from MOOC data</td>
</tr>
<tr>
<td>10/04/2015</td>
<td>Recruitment Letter/Email</td>
<td>Email invitation to students</td>
</tr>
<tr>
<td>10/04/2015</td>
<td>Participant Info Statement</td>
<td>PIS Building User Models from MOOC data</td>
</tr>
<tr>
<td>10/04/2015</td>
<td>Interview Questions</td>
<td>Think aloud interview questions</td>
</tr>
</tbody>
</table>

HREC approval is valid for four (4) years from the approval date stated in this letter and is granted pending the following conditions being met:

**Condition/s of Approval**

- Continuing compliance with the National Statement on Ethical Conduct in Research Involving Humans.
- Provision of an annual report on this research to the Human Research Ethics Committee from the approval date and at the completion of the study. Failure to submit reports will result in withdrawal of ethics approval for the project.
- All serious and unexpected adverse events should be reported to the HREC within 72 hours.
• All unforeseen events that might affect continued ethical acceptability of the project should be reported to the HREC as soon as possible.

• Any changes to the project including changes to research personnel must be approved by the HREC before the research project can proceed.

• Note that for student research projects, a copy of this letter must be included in the candidate’s thesis.

Chief Investigator / Supervisor’s responsibilities:

1. You must retain copies of all signed Consent Forms (if applicable) and provide these to the HREC on request.

2. It is your responsibility to provide a copy of this letter to any internal/external granting agencies if requested.

Please do not hesitate to contact Research Integrity (Human Ethics) should you require further information or clarification.

Yours sincerely

Dr Stephen Assinder
Chair
Human Research Ethics Committee

This HREC is constituted and operates in accordance with the National Health and Medical Research Council’s (NHMRC) National Statement on Ethical Conduct in Human Research (2007), NHMRC and Universities Australia Australian Code for the Responsible Conduct of Research (2007) and the CPMP/ICH Note for Guidance on Good Clinical Practice.
Dear Judith,

Your request to modify the above project submitted on 14 September 2015 was considered by the Executive of the Human Research Ethics Committee at its meeting on 29 September 2015.

The Committee had no ethical objections to the modification/s and has approved the project to proceed.

Details of the approval are as follows:

<table>
<thead>
<tr>
<th>Project No.:</th>
<th>2015/277</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Title:</td>
<td>Building user models from MOOC data</td>
</tr>
<tr>
<td>Approved Documents:</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Type</td>
</tr>
<tr>
<td>14/09/2015</td>
<td>Recruitment Letter/Email</td>
</tr>
</tbody>
</table>

Please do not hesitate to contact Research Integrity (Human Ethics) should you require further information or clarification.

Yours sincerely,

Dr Fiona Gill
Chair
Human Research Ethics Committee
This HREC is constituted and operates in accordance with the National Health and Medical Research Council’s (NHMRC) National Statement on Ethical Conduct in Human Research (2007), NHMRC and Universities Australia Australian Code for the Responsible Conduct of Research (2007) and the CPMP/ICH Note for Guidance on Good Clinical Practice.
Appendix G

Think-Aloud Study: Personal Information Statement
Personal Information Statement

This appendix shows the Personal Information Statement approved by the ethics committee for participants in the Chapter 6 study.
PARTICIPANT INFORMATION STATEMENT

(1) **What is this study about?**

You are invited to take part in a research study about observing student activity in an online course, estimating student knowledge from that activity, and the usefulness of providing an interface to allow students to interact with the system's "model" of themselves to facilitate additional learning.

You have been invited to participate in this study because you are participating in the online course on which the study is based. This Participant Information Statement tells you about the research study. Knowing what is involved will help you decide if you want to take part in the research. Please read this sheet carefully and ask questions about anything that you don't understand or want to know more about.

Participation in this research study is voluntary.

By giving consent to take part in this study you are telling us that you:

- Understand what you have read.
- Agree to take part in the research study as outlined below.
- Agree to the use of your personal information as described.

You will be given a copy of this Participant Information Statement to keep.

(2) **Who is running the study?**

The study is being carried out by the following researchers:

- Professor Judy Kay, School of Information Technology, University of Sydney