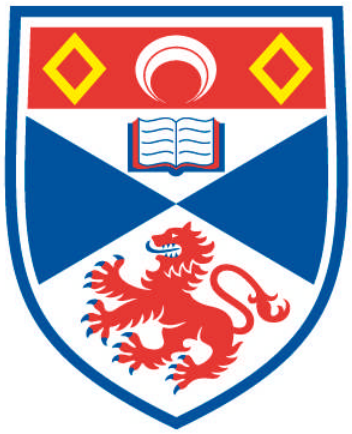
A WEB-ORIENTED FRAMEWORK FOR THE DEVELOPMENT AND DEPLOYMENT OF ACADEMIC FACING ADMINISTRATIVE TOOLS AND SERVICES

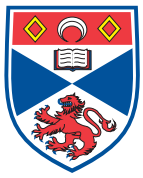
J. Ross Nicoll

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



2015

A Web-oriented Framework for the Development and Deployment of Academic Facing Administrative Tools and Services



A thesis to be submitted to the
UNIVERSITY OF ST ANDREWS
for the degree of
DOCTOR OF PHILOSOPHY

by

J. Ross Nicoll

School of Computer Science
University of St Andrews

Abstract

The demand for higher education has increased dramatically in the last decade. At the same time, institutions have faced continual pressure to reduce costs and increase quality of education, while delivering that education to greater numbers of students. The introduction of software systems such as virtual learning environments, online learning resources and centralised student record systems has become routine in attempts to address these demands. However, these approaches suffer from a variety of limitations:

- They do not take all stakeholders' needs into account.
- They do not seek to reduce administrative overheads in academic processes.
- They do not reflect institution-specific academic policies.
- They do not integrate readily with other information systems.
- They are not capable of adequately modelling the complex authorisation roles and organisational structure of a real institution.
- They are not well suited to rapidly changing policies and requirements.
- Their implementation is not informed by sound software engineering practises or data architecture design.

Crucially, as a consequence of these drawbacks such systems can increase administrative workload for academic staff.

This thesis describes the research, development and deployment of a system which seeks to address these limitations, the Module Management System (MMS). MMS is a collaborative web application targeted at streamlining and minimising administrative tasks. MMS

encapsulates a number of user-facing tools for tasks including coursework submission and marking, tutorial attendance tracking, exam mark recording and final grade calculation.

These tools are supported by a framework which acts as a form of “university operating system”. This framework provides a number of different services including an institution abstraction layer, role-based views and privileges, security policy support integration with external systems.

1. Candidate's declarations:

I, J. Ross Nicoll, hereby certify that this thesis, which is approximately 60,500 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for a higher degree.

I was admitted as a research student in 27th September 2005 and as a candidate for the degree of Doctor of Philosophy in 27th September 2008; the higher study for which this is a record was carried out in the University of St Andrews between 2005 and 2013.

Date _____ *signature of candidate* _____

2. Supervior's declarations:

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Date _____ *signature of supervisor* _____

3. Permission for publication:

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any *bona fide* library or research worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. I have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the electronic publication of this thesis:

No embargo on print copy

No embargo on electronic copy

Date _____ *signature of candidate* _____

Date _____ *signature of supervisor* _____

Acknowledgement

An incredible number of students and staff have been involved with the MMS project over the last decade, and attempting to thank everyone who has had an influence on the system is an impossibility. As such these acknowledgements can only hope to address the most important contributions.

First and foremost I would like to thank my supervisor and mentor, Colin Allison, for the motivation, encouragement, and above all else his confidence throughout.

A number of other research staff have contributed directly and indirectly to MMS over the years. The early coursework and attendance tools for MMS were developed by David McKechan and Bin Ling, and their feedback on the API design helped shape its early formative years.

Alex Bain has provided invaluable feedback and encouragement in his role as school administrator for Computer Science, particularly for administrative process analysis.

Stuart Purdie, who joined the team in 2005, has worked tirelessly on many of the tools, as well as the rendering and institution abstraction layers and his contributions to these elements cannot be overstated. I would particularly like to thank him for his constant challenges and feedback.

Victoria Davidson has been key to the support and training for MMS since she joined the team in 2009.

Several students (both undergraduate and postgraduate) have worked on tools for MMS, and their individual contributions to its tool suite have been invaluable in assisting in showing the scope of the project. I would specifically like to thank Kris Getchell (and Alan Miller as his supervisor) for their confidence in MMS and willingness to make it a signif-

icant component of his PhD thesis on LAVA. Thanks go to Clare Walsh (née Kerbey) and Ruth Hardy's for developing these tools for LAVA.

Many thanks must go to a number of open source projects for libraries and other software on which MMS depends (all URLs correct as of 9th May 2013):

- Apache Commons FileUpload[1] provides support for uploading files such as coursework
- Apache Log4J [2] for log reporting
- Apache POI[3] to provide Excel spreadsheet support
- Apache Tomcat[4] as servlet container to host the application
- Bouncy Castle Crypto APIs[5] to provide cryptographic e-mail signing
- Freemarker[6] as a template engine for rendering output
- iText[7] to provide PDF export support
- Java[8] as language and virtual machine
- JFreechart[9] to produce graphs for reporting statistics
- JUnit[10] for unit and integration tests
- Mozilla Rhino[11] to provide scripting support

The icons used within MMS are from the Crystal package (`http://www.everaldo.com/crystal/`, visited 1st March 2008), although the icon original Crystal icon (Crystal SVG) set they are taken from appears to no longer be available.

[1] `http://commons.apache.org/fileupload/`
[2] `http://logging.apache.org/log4j/`
[3] `http://poi.apache.org/`
[4] `http://tomcat.apache.org/`
[5] `http://www.bouncycastle.org/`
[6] `http://freemarker.org/`
[7] `http://www.lowagie.com/iText/` - please note that version 2 is used due to changes in licensing in later versions
[8] `http://www.java.com/`
[9] `http://www.jfree.org/jfreechart/`
[10] `http://www.junit.org/`
[11] `http://www.mozilla.org/rhino/`

Collaborations

MMS is based on work dating back to 1998, and accordingly credit must be given to a great many individuals.

Of the work described in this thesis, the architectural design of MMS is my own work based on experiences working with TAGS' design[12]. The core framework is primarily my own work with the notable exception of the "Flex" table rendering API. In detail:

- The handling of users, modules, groups, and tools, with a design building on previous work and university structures was my work.
- Initial design and development of the roles model which provides the authorisation layer was my work, as was handing of role variability across the institution and the understanding of anti-roles.
- The request handling, routing and validation components were primarily my design and development.
- Moodle integration was my work.
- Re-integration of DIF to provide disability information display was Stuart Purdie's work.
- Display of exceptional circumstance information was Stuart Purdie's work.
- Academic alerts requirements were specified by senior management at the University of St Andrews, however design and development was my work.

[12]TAGS was maintained by David McKechan prior to my involvement with the project, and led by Colin Allison

- The institution abstraction layer was my idea and initial design and development. Later developments by Stuart Purdie added integration with SITS:Vision and data source weighting/confidence management.

Of the integrations with external systems:

- Data Warehouse, SITS:Vision and advisor's database integrations have been worked on by myself and Stuart Purdie.
- Integration with Talis Aspire (reading lists) is my own design and development.
- Integration with the EvaSys Education Survey Automation Suite is my own design and development.
- Integration with virtual worlds was my idea, design and development.

Of the tools hosted within MMS:

- The coursework tool was originally written by David McKechan, but has been very extensively rewritten since then by Stuart Purdie and myself.
- The exam and final grade tools were designed and developed primarily by Stuart Purdie.
- The file share and notebook tools were initially derived from TAGS' design, but revised design and all development was my work.
- The design and development of the quiz tool was my own work, based on suggestions provided by the Department of Film Studies and Clare Kerbey.
- The student group sign-up tool was Stuart Purdie's design and development based on requests from administrative support.
- Academic Alerts emerged from a policy review lead by the VP: Teaching and Learning and Deans. The refinement, implementation and deployment was my work.
- The LAVA archaeological dig tools were others idea and development, and are presented merely to illustrate uses of MMS.

- Postgraduate research student admin tools were Colin Allison's idea, with my design and development.
- School/unit-level tools for bulk management of administrative tasks (for example student/staff on modules) was my design and development.
- Coversheet generation support was my design and development.
- The Word document handling framework was my own design and development, including conversion to e-book format.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis considers firstly "Can academic administrative overheads can be reduced through the use of web applications?" The underlying work has been iterative, and further questions will be addressed through the course of the work.

Given the recent and continued increases in demand for student places at higher education institutions[58][45], there is a clear need to reduce overheads in higher education teaching, to enable academic staff time to be devoted primarily to teaching and research tasks. Reduction and changes in higher education funding[48] provide motivation for increased competition in the higher education sector as a whole.

Increases in administrative costs in higher education are well documented[41][40] [69][1]. Jürgen also reports that over 40% of professors' time (in England, the Netherlands and Sweden) is spent on administration[27]. This illustrates a clear opportunity for reducing costs through minimising academic time spent on administrative tasks, enabling a better return on investment.

The preceding TAGS[5] and INSIDE[21] systems had proven that web applications have scope for use in administration, however their designs required major work in order to extend significantly further.

I will describe the design and implementation of a web-based application suite, the Module Management System[1][2] (MMS), which is in active use across the University of St

[1]Note the Greene, Kisida and Mills report is not peer-reviewed

Andrews since 2010. MMS provides an inter-connected set of tools for common administrative tasks, seeking to both streamline these tasks and reduce risk of mistakes. These tools are enabled by a software framework which tackles a number of challenges:

- Institutional data models are extremely complex. As an example the SITS:Vision student record system alone contains well over 2,000 inter-related data tables.
- Institutional data may lack a single "golden source" in some cases, instead requiring that multiple data sources must be examined in order to ensure the most accurate data is captured. These problems typically arise where individual software systems lack the capability to accurately model the data.
- Institution policies may vary significantly not just between institutions but also within a single institution.
- Policies and requirements are frequently revised or corrected during an academic year, requiring frequent changes to implementation, sometimes against extremely limited timescales.
- Data may be edited concurrently and independently by a large number of users, and this can lead to confusion with regards to the origins of changes to data. This in turn can result in users mis-attributing changes, and in some cases to assume that a change is caused by a software error, which undermines confidence in software systems.
- Security must be enforced throughout the system, in line with institutional policies.

In their 2013 paper White and Davis describe an institutional benefits framework used to analyse benefits a project will provide[134]. In that context, MMS' objectives are focused on the impact areas "Institutional Benefit", "Institutional Indicators" and "Faculty Benefit", reducing costs to the institution by decreasing workload on staff, while reducing risks in data management. Further impact areas "Student Learning" and "Student Experience" also benefit from the streamlining of administrative processes, but this is a side-effect rather than the primary intention of the software.

Given developer and support staff resources are direct costs to the institution, staff resourcing requirements must be highly controlled in order for a system to provide an overall cost reduction. This limitation means that naively using more developers, testers or support staff

to resolve other requirements is counter-productive to the overall goals. Accordingly there are challenges in minimising expenditure while still achieving the project goals.

MMS is designed to support a set of complementary approaches to reducing administrative overheads in a higher education environment. These approaches are, in brief: correcting issues in data quality, reducing risks to data quality, exposing centrally maintained data to end users in a useful way, emphasising currency of data provided to end-users, providing user tools for reviewing and managing processes, proactively responding to data issues, and minimising user interaction with administrative software.

This thesis will examine the background context from which MMS was designed, the challenges encountered in both the development and roll-out of MMS, the design principles on which MMS is based, how the challenges have been addressed by the framework and user tools, and lastly evaluate the system's design and success in reducing overheads.

MMS at the University of St Andrews currently has over 14,000 student users[2] and 5,800 staff users, and receives 60,000–80,000 HTTP requests during a typical weekday.

1.1 Motivation

The datasets used in administration are inherently shared between multiple members of staff, and in many cases directly involve the students the data relates to (for example coursework submission and return of assessment outcomes). Further, administrative workflows frequently require multiple staff working together to complete a task. As such collaborative tools are an obvious fit for the use-case.

Dedicated student record systems such as SITS:Vision[3] are adequate for modelling the actual structure of student data, but are ill-suited for use by students and academic staff due to their complexity, typically requiring specialised training to use effectively. Student record systems may also require the use of application-specific client software, as is the case with SITS:Vision, adding further obstacles to widespread usage of the systems.

[2]This figure includes a number of alumni, visiting students, etc. and is therefore significantly higher than the number of full time students

[3]`http://www.tribalgroup.com/technology/sitsvision/Pages/default.aspx` - accessed 28th May 2103

A web interface was chosen to provide a platform which is widely available to end-users, well understood, trivial to update and asynchronous. In comparison to obvious alternatives such as a dedicated desktop application, the interface design is inherently limited in its flexibility (such as range of form elements), however this is considered a worthwhile trade-off for the near-universal availability of compatible computers. The relatively recent HTML 5 specification significantly mitigates these limitations.

Data and workflow management tasks are chosen as key improvement targets because both were partially addressed by TAGS and INSIDE. This experience suggested a number of possible further improvements to the application designs, which led to the design of MMS.

1.2 Background

MMS is a major step forward, building on previous research projects at St Andrews, namely Tutors and Groups Support[5] (TAGS) and Institutionally Secure Integrated Data Environment[21] (INSIDE), and addressing emerging requirements from the academic staff at university. MMS incorporates lessons learnt from both of these projects, in its end-user functionality, however its core framework has been completely redesigned and rewritten.

MMS was designed to overcome architectural, functional, security, development and performance limitations inherent in the design of the TAGS framework, while adopting best practises from INSIDE's integration with core institutional record systems. In doing so MMS blends elements from virtual learning environments (VLEs) with student records systems and web portals. MMS is intrinsically staff-centric in its designs, with student-facing functionality present only as needed to achieve improvements for academic staff.

Development of MMS started in the summer of 2002, and went live within the School of Computer Science in September of the same year, in time for the start of the academic year 2002/3. Use of the INSIDE proof-of-concept software was discontinued ahead of 2002/3 as MMS acted as the production edition, while TAGS remains in-use for a number of teaching-centric projects at other institutions. A brief history of projects prior to MMS, along with the development timeline of MMS, is shown in figure 1.1).

For the initial three years, work on MMS focused on student management functionality.

Adoption of MMS was solely driven by academics opting to use it within their modules, and as such as it had to quickly show value.

Date	Event
1999	TAGS project begins
2000	INSIDE project begins
2001	TAGS ported from from Tcl to Java
2002	Development of MMS begins, MMS goes live within the School of Computer Science
2002	INSIDE proof-of-concept software obsoleted by MMS
2003	MMS adopted by the School of Mathematics
2003	Disability Information Flow (DIF) tool added as a sub-component of MMS
2004	DIF split off as a separate application
2005	Start of work covered by this thesis
2007	MMS deployed onto hardware managed by Library and Information Services
2007	DIF replaced by SITS:Vision based application
2007	MMS goes live in the Schools of Economics, Philosophy, and Psychology
2009	MMS required to be used in reporting all module grades
2010	MMS and Moodle replace WebCT for institutional virtual learning environment. DIF re-integrated into MMS in response to demand from academic staff

Figure 1.1: Time-line of MMS Development

MMS today has over a dozen different tools that can be made available through each taught course module, historical data back to the academic year 2005/6, and over 13,000 users in its database. Since the academic year 2009/0 it is a required part of reporting final grades from every module in the university, and as such used in some capacity by every teaching unit.

The abstract raised the following drawbacks with current approaches to conventional VLEs and student record systems:

- They do not take all stakeholders' needs into account.

- They do not seek to reduce administrative overheads in academic processes.
- They do not reflect institution-specific academic policies.
- They do not integrate readily with other information systems.
- They are not capable of adequately modelling the complex authorisation roles and organisational structure of a real institution.
- They are not well suited to rapidly changing policies and requirements.
- Their implementation is not informed by sound software engineering practises or data architecture design.

By comparison, this thesis shows how MMS addresses these points:

- MMS takes all stakeholders' needs into account, as is evidenced by the design based on feedback from not just academics, but also students, administrative staff and management. This is discussed further in section 4.3 of the design chapter.
- MMS seeks to reduce administrative overheads in academic processes, though improved management of data. Specifically the integration with other information systems both minimises configuration, and enables data held in MMS to be transferred to other systems without requiring manual intervention or migration (for example the final grade reporting tool described in 7.5). The data architecture is designed to minimise risks through normalisation and data integrity constraints, and audits most data changes over time in order to establish clear detail on how and when changes have been made, (see chapter 4, section 4.8). The effectiveness in reducing these overheads is the basis of the evaluation in chapter 11.
- MMS reflects academic policies, such as Academics Alerts (see chapter 8), Final Grade reporting (see chapter 7, sections 7.6.1 and 7.5), postgraduate research examination panel nomination and annual reports to the Pro Deans (chapter 9).
- MMS integrates with a number of information systems as is evidenced by its interactions with systems including SITS:Vision, the institution's data warehouse, Moodle, and Google Search Appliance, covered in chapter 6.

- MMS successfully models the complex authorisation roles and organisational structure of the institutions, as described in chapter 5, section 5.7.
- MMS is well suited to rapidly changing policies and requirements, through use of approaches including resilient architectural design and user-facing scripting tools. The benefits of this can be seen in the ability for users to manage school-level policies on lateness, final grade calculation and marking scales themselves (see section 7.6). This also enables rapid change, as illustrated with Academic Alerts (chapter 8), the move to Moodle as institutional VLE (the impact of which on MMS' requirements is described in section 7.7) and development of LAVA (chapter 10).
- MMS implementation is informed by sound software engineering practises and data architecture design as described in the Architecture chapter. See sections 5.3 and 5.4 for adaptation of standard design patterns, section 5.13 for isolation of concerns, and sections 5.5 and 5.8 for novel work in software design. Data architecture (both in the database and as files on disk) is described in the design chapter, section 4.8.

1.3 Structure of Thesis

General context of the project is provided in chapter 2 (Context), which describes a number of applications to illustrate other work in the area, as well as differentiate MMS from that work. Challenges (chapter 3) examines in detail the difficulties developing software in higher education, which must be overcome to enable the tools which form the user-facing elements of MMS.

Chapters 4 and 5 (Design and Software Architecture) look at how these challenges were solved. Design describes the theory of MMS' design and the principles it is based on, while Software Architecture looks at implementation of the framework based on this design which supports the embedded tools described later. The design and software architecture pair with the challenges described earlier in the thesis, and contain a number of novel elements which underpin to MMS' success (data version control, institution modelling, inclusion of administrative staff in design, server-side scripting, etc.)

Chapter 6 (Integrations) looks at third party applications with which MMS interoperates, both as part of the framework and the user tools.

Chapter 7 (User Tools) introduces the user-facing tools which deliver MMS' core functionality, and are the main elements of functionality used to reduce administrative overheads. Other tools are also provided with the intention of encouraging use of MMS and improving the user experience, especially those tools more orientated towards learning and teaching. A number of the tools described are specific to MMS or have unique features and form part of the novel work of this thesis.

Howing described MMS, this is followed by three case studies (chapters 8 to 10) looking at the use of MMS as an academic policy implementation tool (Academic Alerts), a workflow management tool (Postgraduate Research Student Administration), as and as a research framework (LAVA). Each of these chapters incorporates elements of evaluation.

Evaluation of MMS (in chapter 11) is performed in terms of usage statistics over time, and surveys of the users will be used to assess acceptance of MMS. A summary of the main successes and failures is included at the end of this chapter.

Lastly, future work (in chapter 12) reflects on underdeveloped ideas from MMS' development as well as describing a number of ideas for further architectural and functionality extensions. This chapter also looks at lessons learnt from MMS which could be readily applied to other applications.

Chapter 2

Context and Related Work

The context for MMS is easily introduced by the substantial number of web applications designed for improving higher education. Virtual learning environments (VLEs) are now commonplace, and comprise a large proportion of these applications. Examples such as ANGEL Learning Management Suite, Blackboard Learn[1], Bodington[2], Moodle[3], Sakai CLE[4] and WebCT provide a variety of tools and a framework to support them, however they are all primarily focused on improving learning effectiveness.

In contrast MMS provides a very different range of tools aimed at module administration, with learning and teaching as a secondary goal. It also seeks to facilitate development of such tools, simplifying the development process for others to be able to readily add further functionality.

There are some similarities to the BOSS[5], and uPortal[6] applications. BOSS is an advanced coursework submission tool with automated marking functionality, and uPortal is a web portal designed to collate and present information and tools to students and staff. Both of these reflect some of the functionality of MMS and accordingly a detailed comparison is presented later in this chapter.

[1] `http://www.blackboard.com/Platforms/Learn/Products/Blackboard-Learn.aspx` - accessed 13th November 2013

[2] `http://sourceforge.net/projects/bodington/` - accessed 13th November 2013

[3] `https://moodle.org/` - accessed 13th November 2013

[4] `http://www.sakaiproject.org/` - Accessed 13th November 2013

[5] `http://www.dcs.warwick.ac.uk/boss/index.php`

[6] `http://www.jasig.org/uportal`

This chapter will look at VLEs, other related web applications, architecture and API design, and end with a quick introduction to the preceding applications at St Andrews which influenced MMS' design. A broad summary of comparable applications is shown in table 2.1, and as can be clearly seen there is a trend towards shrinking choice over time, with few new applications being launched, and those projects which continue being merged under single organisations.

Application	Category	Publisher	Still Active
ANGEL Learning	VLE	Blackboard	No
Blackboard Learn	VLE	Blackboard	Yes
Bodington	VLE	University of Leeds	No
BOSS	Coursework Submission	University of Warwick	Yes
Moodle	VLE	Moodle Pty Ltd	Yes
Sakai	VLE	Apereo Foundation	Yes
SITS:Vision	Student Records	Tribal	Yes
uPortal	Web Portal	Apereo Foundation	Yes
WebCT	VLE	Blackboard	Yes

Table 2.1: Summary of Related Web Applications

2.1 Virtual Learning Environments

Given the nature of MMS as a student-facing web application, comparisons with virtual learning environments (VLEs) such as Blackboard Learn[7], Moodle[8] and Sakai CLE[9] are common. VLEs have seen widespread adoption across higher education[57][18], and as such are frequently encountered by users, who commonly infer similarity in some of the tools provided by these systems to mean that the applications are intended for similar purposes.

However VLEs focus on improving the delivery of teaching, and can actually increase workload on academic staff[47][91][113]. In contrast, MMS focuses on improving admin-

[7] `http://www.blackboard.com/Platforms/Learn/Products/Blackboard-Learn.aspx` - accessed 6th June 2013

[8] `https://moodle.org/` - accessed 6th June 2013

[9] `http://www.sakaiproject.org/sakai-cle` - accessed 6th June 2013

istration, and has teaching-related functionality only in terms of this goal, rather than as a primary feature.

Since this thesis was started, Blackboard Inc. have acquired two of the key VLEs covered in this context survey (ANGEL and WebCT), and these two platforms are being phased out in preference of Blackboard Learn.

Additionally Bodington (initially developed at the University of Leeds) has since been replaced by other VLEs (Blackboard at Leeds, Sakai at Oxford).

This leaves the marketplace substantially slimmer than when this thesis was started (no specific conclusion is intended about this fact). These historic projects are included below for reference.

2.1.1 Blackboard Learn

Blackboard Learn is a widely used commercial virtual learning environment by Blackboard Inc. Learn provides a wide range of tools for content delivery, online assessment and collaboration. Blackboard Inc. have previously purchased WebCT[52] and ANGEL Learning[53], and accordingly Learn has functionality taken from both of these software packages, making it an obvious choice for institutions moving from ANGEL Learning or WebCT.

Course-level tools provided by Learn include assignment submission and grading, calendar, course content management, discussion boards, notifications, wiki.

Building Blocks

One interesting aspect of Learn is its plugin support. Whereas MMS provides some extension capability through a scripting environment within the MMS itself, Learn acts as a web application container for plugins. These plugins are referred to as “Building Blocks”.

This is quite different to MMS’ scripting support in implementation and intention. Plugins are installed system-wide and are extensions to Learn’s functionality, whereas MMS’ scripts are typically managed departmentally or even on a per-module basis, and provide

simple calculations.

As a technology however it is interesting, and suggests a possible approach to modularising MMS in future (through managed, hosted applications).

2.1.2 Moodle

Moodle is an open source course management system, which provides a selection of tools use in delivering course content and assessments to students. Beyond simple content management, it provides quizzes, discussion forums, event tracking, Wikis, and analysis tools for monitoring academic performance. Commercial hosting and support options are available from a variety of companies, but the project itself is open source and free to use.

Example screenshots of Moodle are shown in figure 2.1.

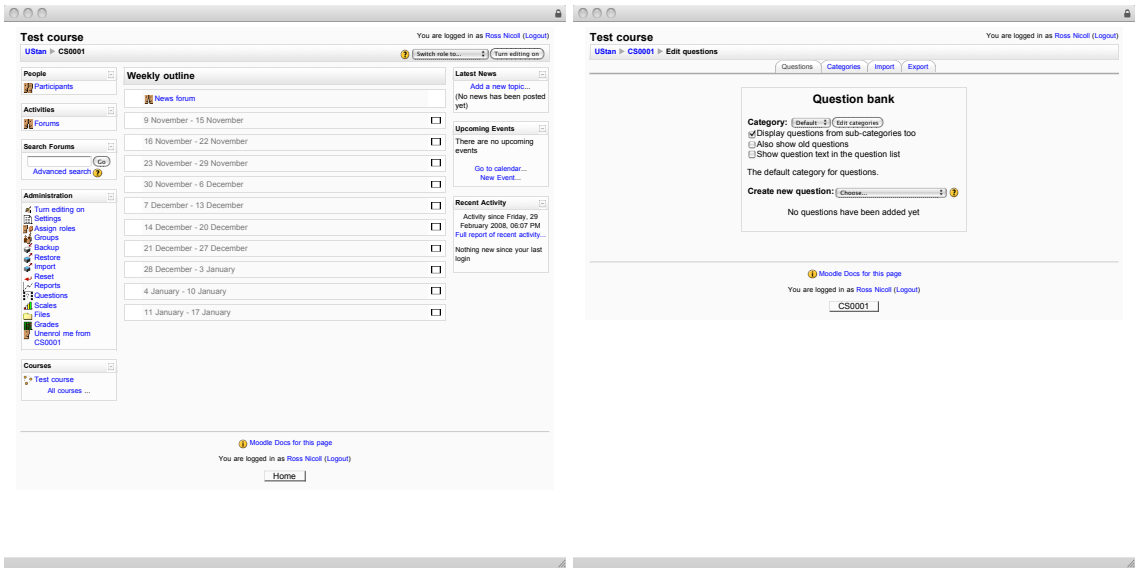


Figure 2.1: Screenshots of Moodle

Moodle has proven extremely popular, with tens of thousands of registered Moodle installations worldwide, including over 4,000 in the UK alone. During the writing of this thesis, the total number of registered installations increased from around 35,000 in 2008, to over 82,000 as at 26th June 2013[85].

Moodle replaced WebCT (see also 2.1.5) as the standard VLE for the University of St Andrews as of the academic year 2010/1.

Unlike MMS, Moodle is written in PHP. Its database layer was designed based on functionality of MySQL 3.x[50] and therefore lacks foreign key constraints, leading to significant numbers of constraint violations[108]. The database structure, while significantly improved over earlier versions, still embeds complex data types in single fields, for example the "displayoptions" field of the "resource" table (this example based on Moodle 2.6):

```
a:2:{s:12:"printheading";i:0;s:10:"printintro";i:1;}
```

This leads to data which is hard to interpret or manipulate without reference to the application code, as well as requiring that any changes to any item of data affect all contained data within the same field. This potentially could cause unexpected results if two users are manipulating the same data concurrently, as one user could overwrite changes made by another depending on the database locking model.

2.1.3 Sakai CLE

Sakai CLE (Collaboration and Learning Environment) is an open source VLE developed by the Sakai Foundation, which recently merged with JASIG (who manage uPortal) to form the Apereo Foundation[30].

Sakai CLE not only provides conventional learning management functionality, but its tools can also be used for research and project collaboration and digital portfolios of work.

Key points of similarity with MMS include support for coursework assignments, content management tools, and a grade book for storing grades (however this does not handle transfer of data back to central records).

2.1.4 ANGEL Learning Management Suite

ANGEL (A New Global Environment for Learning) Learning Management Suite is a virtual learning environment, based on the OnCourse system developed by Dr. Ali Jafari at Indiana University[54][55]. In 2000, Dr. Jafari formed CyberLearning Labs, Inc.[80] with support from Indiana University's Advanced Research and Technology Institute (ARTI) to develop a virtual learning environment, which became ANGEL Learning.

In May 2009, ANGEL Learning was acquired by Blackboard, who currently sell the system under the name "Blackboard Learn ANGEL Edition[10]." Blackboard have also added functionality to their core Learn product, based on functionality provided by ANGEL.

2.1.5 WebCT

WebCT is a virtual learning environment, developed by Murray W. Goldberg in 1996 at the University of British Columbia[38][37]. It is of particular note as it was the original VLE used at the University of St Andrews[131] for delivery of course content.

WebCT provided a variety of tools, including:

Announcements Sends announcements of course-related events to students and staff.

Assignments Provides details of assignments to students, and provides both space for them to submit coursework to, and to hold marks for that coursework.

Calendar Provides a calendar of events, handling both as shared events for an entire course, and an individual user.

Chat and Whiteboard Provides for users with a method for synchronous communication (chat room), as well as a whiteboard for sharing visual data such as slides.

Course Content Stores course content in a file-system structure, ready for access by students. Course content can also be played back from SCORM modules.

Discussion Provides for asynchronous communication about a course, as well as assignment of marks to conversations.

HTML Creator A WYSIWYG (What You See Is What You Get) HTML editor, allowing non-technical users to compose HTML pages without requiring them to learn HTML first.

Learning Modules Sequences course components, such as content, assessments, assignments, media library components, SCORM modules, chat rooms, etc. to provide a learning path for students to follow.

[10] `http://www.blackboard.com/Platforms/Learn/Products/Blackboard-Learn/ANGEL-Edition.aspx` - accessed 8th July 2013

Mail Provides a secure e-mail facility for students and staff to communicate privately with each other. However e-mail can only be received from other users of the system.

Media Library The Media Library is a collection of pictures, video files, audio files and/or glossary entries.

Quizzes WebCT provides for a wide variety of different tests to be administered to students, to evaluate how well they have understood course material. Some of these tests can be automatically marked (multiple choice, calculated, fill in the blank) while others are manually marked by course staff (paragraph/essay-style).

Self Tests Similar to quizzes, except intended to allows students to assess their own knowledge of course material, rather than for assessment by teaching staff.

Surveys Provides a way of gathering feedback about the course, in an anonymous manner, using a series of staff-defined questions.

2.1.6 Bodington

Bodington was originally developed by the University of Leeds. It proceeded to expand to a number of other institutions including the University of Oxford, University of Manchester, University of York, UHI Millennium Institute and Eton College.

As of release 2.6.0 Bodington provided a substantial set of course management features:

Group communication room A discussion space similar to TAGS' notebook service

Logbook For students for keep notes on their coursework

Content management For course content

Questionnaire For performing surveys of the user base

Simple tests With options for either automatic or manual marking

Pigeon holes For coursework submission by students

These features are however all intended for use in assisting learning, and are student-focused rather than administrative.

Note that the last released version as of the time of writing is 2.8.0[11], however documentation is difficult to acquire due to the main project website no longer being available, and therefore a historical list has been used.

2.2 Other Software

2.2.1 BOSS

BOSS[12] (BOSS Online Submission System) is a coursework submission and marking tool developed at the University of Warwick[60]. Unusually for the software systems detailed here it focuses on providing a single service, namely electronic submission of coursework, and assessment of that coursework. The functionality it provides is similar to that of the coursework submission tool included in MMS, although BOSS expands upon that functionality by providing tools for automatic marking of coursework. However BOSS appears to lack integration with central student records.

Evaluation of the BOSS system has shown the use of electronic mark sheets as an effective tool in reducing time taken for marking[61], supporting the theory that web applications can reduce time taken in other administrative tasks.

From a technical point of view it is also interesting in that BOSS has support for alternative user interfaces – as well as a web interface, it comes with more traditional desktop application clients for students and staff. It does this by splitting the application into two parts, a core ”engine” and the user interface, which communicate using Java remote method invocation (RMI).

[11] `http://sourceforge.net/projects/bodington/files/bodington/2.8.0/` - accessed 14th May 2013

[12] `http://www.dcs.warwick.ac.uk/boss/` - accessed 9th May 2013

2.2.2 Mobile Applications

Mobile applications targeting higher education environments frequently include functionality for delivering academic-related data to students. Examples include campusM[13] from oMbiel, u360 Mobile[14] from Straxis Technology and uMobile[15] from Jasig.

While there are no generally available mobile applications for MMS, a number of different applications have been prototyped[73][125]. Please note that web applications with an alternative rendering specifically for mobile platforms are not covered in this section, instead this only addresses "native" applications.

Stand-alone Mobile Applications

Stand-alone applications such as campusM and u360 Mobile (targeting the UK and US higher education markets respectively) have the advantage that they do not require the institution to have a specific software package (such as Moodle, Blackboard Learn or uPortal) to use, however they also require more work to integrate with such platforms (if even possible) and institutional data sources.

The majority of the functionality presented by these applications closely reflects content directly available from an institutional website, such as a staff directory, library catalogue search, news feeds, calendar of events, course catalogue, etc. Where they do provide functionality that does depend on or integrate well with mobile platforms such as campus maps and alert (push) messaging, this functionality does not overlap with functionality provided by MMS.

By comparison mobile application prototypes for MMS have focused on dealing with time and/or location sensitive information such as grades received, tutorial group signup and attendance monitoring.

[13] `http://www.campusm.com/` - accessed 11th June 2013
[14] `http://www.u360mobile.com/` - accessed 11th June 2013
[15] `http://www.jasig.org/umobile` - accessed 11th June 2013

Integrated Mobile Applications

Mobile applications which integrate with a specific piece of software have the advantage of a data service whose development is interwoven with that of the application. The parent application is likely to already be installed and well established at the institution, meaning that data quality and availability are much less likely to be problems. However, this also inherently makes these applications inflexible, dependent on functionality of their host application.

uMobile, which integrates with the uPortal portal platform, is based on the Appcelerator Titanium SDK[16]. This provides a flexible and powerful framework, including mobile-orientated functionality such as maps, location and push notifications, however its included default features (calendar, news, video, staff directory, search and maps) do not overlap with MMS.

Moodle Mobile[17] is based on the Phonegap framework[18], although an earlier version was a "native" iOS application. It provides a mobile-optimised interface to some of Moodle's functionality, such as course content, or messaging students. The functionality is teaching-centric, and has limited overlap with MMS (for example it does allow access to course content, but doesn't appear to provide grade data).

2.2.3 SITS

The SITS:Vision student and course management system, developed by Tribal, is one of the most popular student management systems in the UK, and is the system of choice at the University of St Andrews. The data it manages overlaps heavily with the data maintained within MMS, including student and staff details, taught modules, module enrolments, students' final grades from modules, etc. There are significant data elements it stores which MMS does not, such as degree courses, and details of potential students, as well as elements that it could store but does not at St Andrews, such as marks or grades for individual items of assessed work.

[16] `http://docs.appcelerator.com/titanium/latest/` - accessed 22nd February 2015
[17] `http://docs.moodle.org/dev/Moodle_Mobile` - accessed 15th July 2013
[18] `http://phonegap.com/` - accessed 15th July 2013

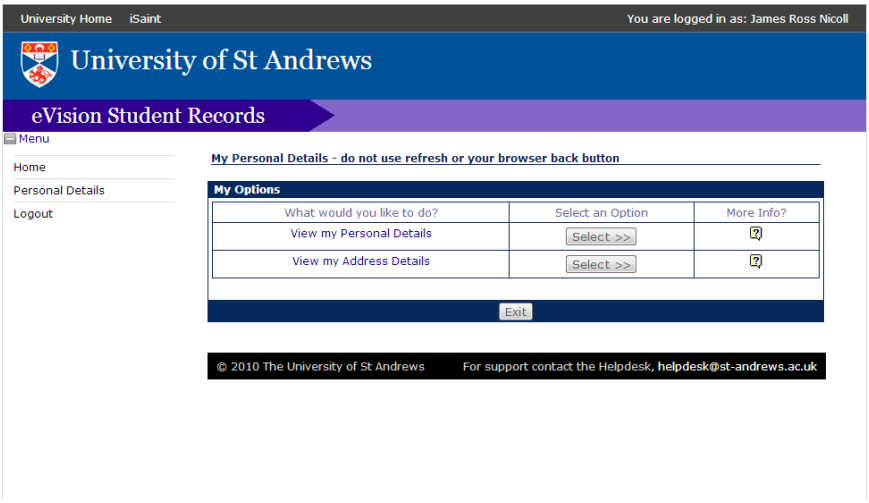


Figure 2.2: Screenshot of e-Vision

Unlike MMS, it is a desktop application intended primarily for use by expert users. It has a web portal interface, e-Vision (shown in figure 2.2), designed for a more general user-base, however the workflows presented by e-Vision are generally constrained by the underlying data structures. For example users can generally edit a single record at a time, and workflow during editing is tightly controlled (note instructions not to refresh or use the back button, in the screenshot).

The underlying data architecture of SITS:Vision is intended to be database agnostic, such that while it happens to use Oracle as the database engine at St Andrews, virtually any other SQL-based database can be used instead if desired. Unfortunately it also fails to use many core database features, for example foreign key constraints on data tables, or primary key generation by the database, leading to unnecessary risk of data corruption. Further, actions such as triggers on data change are managed in the application software rather than the database layer, which mean that other applications cannot safely modify SITS' data within the underlying database, and essentially duplicates engineering effort in ensuring that triggers are fired correctly every time data changes.

2.2.4 uPortal

uPortal[19] is interesting as a further example of a web application intended for use in an educational context, which is not focused on delivery or administration of a course. uPortal is, as the name suggests, a web portal application, which aggregates interfaces and data from a number of sources to provide a central location for students and staff to find webmail, news feeds, calendar, etc.

It is in use at a large number of institutions, including the University of St Andrews, where it provides functionality including matriculation, checking personal details, and links to coursework and modules within both Moodle and MMS.

uPortal has similarities to MMS in that both pull together data from a wide variety of sources, however uPortal takes data which has already been formatted as HTML (through portlets), whereas MMS uses the raw data and handles presentation itself. Here the intents are different, and while uPortal is simpler, it lacks the ability to readily mix data sets together (for example merging students on modules with calendars).

2.3 Architecture

While the most obvious aspects of MMS are its end user functionality, its architecture is also key. MMS provides a web application framework, which simplifies development of tools for use in a higher education context. This framework encourages effective design of tools, while providing support for authentication and authorisation. It enables use of a wide variety of institutional data sources and systems, through an abstraction layer which manages much of the complexity for the tool developers.

2.3.1 Model-View-Controller

MMS' web interface is derived from the Model-View-Controller (MVC) pattern[65], which it encourages in its hosted tools through provision of a data model, and easy tools for separating the controller and view. The main modification to the MVC model as implemented

[19] `http://www.uportal.org/` - accessed 9th May 2013

in MMS is the addition of a layer which handles security, particularly in relation to access controls to entities within MMS (such as modules, course content, students, etc.)

2.3.2 Spring

In the context of Java application architecture, especially MVC web applications, the Spring framework is an obvious comparison. Spring is a framework based around use of aspect-orientated programming (see Kiczales et. al.[64] for an introduction to aspect oriented programming) to modify Java Beans, injecting significant new functionality such as transaction handling, security, web application support, etc.

Like MMS, Spring provides a framework which includes web request dispatching and parsing support, database querying and manipulation utility classes, model-view-controller support, web services, etc. It differs from MMS in that it provides a generic framework, whereas MMS is targeted at dealing with data structures in a higher education institution.

Whereas Spring uses aspect-orientated programming to provide much of its functionality, MMS acts as a wrapping layer which directly calls its hosted tools. This approach ensures that flow of control is clearer as it is limited to utilising conventional Java language features, however it does so at the cost of requiring hosted tools to be designed for the framework. In comparison, Spring uses proxy classes to modify behaviour of existing classes, leading to non-obvious flow of control. The number of distinct method calls used by Spring as part of this process also leads to stack traces which have a high proportion of framework lines compared to the hosted application[94], further confusing the process of debugging errors.

2.4 Project Background

The initial design of MMS was formed from experiences with a variety of applications. These applications allowed experimentation with different designs, use cases, development languages etc., and from them the architecture and intent of the first version of MMS was formed.

Of the applications listed below, Tutors and Group Support and Finesse are both still in use in finance education, INSIDE was replaced entirely by MMS, and DIF (which was

actually a sister-project of MMS, not a predecessor and ran from 2004 to 2007) has since been merged into MMS.

2.4.1 Tutors and Group Support

TAGS[5] was a Scottish Higher Education Funding Council (SHEFC)[20] funded project to research, develop and deploy networked learning environments. The TAGS project was started in 1999. It was developed in the Tcl scripting language, using the cgi.tcl[70] extension for web support, although later additions also used Perl and Python in parts, and it was finally re-written in Java during 2001-2002[90]. The core TAGS framework provided authentication (against its own user database), and simple authorisation based on a model based on users, groups and resources.

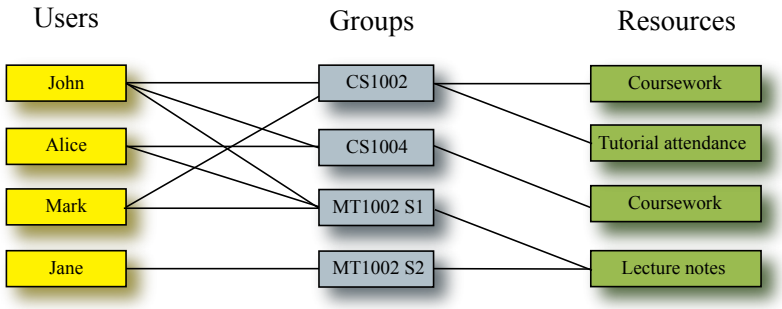


Figure 2.3: TAGS: Users, Groups and Resources Model

Under this model (shown in figure 2.3), users and resources could be assigned to groups in a many-many relationship. When resources are assigned to groups, they can be set as possessing read and/or write access privileges. Users then have the union of accesses provided to each resource, by all groups they belong to. It rapidly became apparent that as TAGS was used in several distinct institutions, the number of users, groups and resources visible to staff was overwhelming, and a new structural unit was required in order to model institutions. These were referred to as “domains”. Users, groups and resources could all be assigned to domains in a many-many relationship, although most were assigned to a single domain, with exceptions for cases such as system administrators who actually operated across multiple institutions.

TAGS provided a number of tools for supporting course administration:

[20]Where were replaced by the Scottish Funding Council in 2005

Archival Tool A management tool which can create archive files of static HTML representing the content of a group, including both users and resources. An archive could be downloaded to a local disk or kept online. All the components of an archive, including user IDs and resource instances, could then be deleted from the TAGS system without losing the details of a particular class and their work. This allowed for recycling of groups across multiple academic years as a method of mitigating clutter.

Document Approval Tool Provides coursework spaces into which students can upload work. Staff can then mark that coursework and/or provide feedback on it. Staff can be automatically notified by e-mail when students upload coursework, and students in turn can be notified when their work is marked.

Protected Page Set Wraps a set of files within the TAGS security infrastructure so that users must log in to access the files.

Reflective Marksheet A tool for students to reflect on their own work, with space for feedback from tutors.

Tasklist Extends the protected page set to provide a sequenced set of page for students to read and/or complete work within.

Tutorial Attendance and Performance Register Holds records of student attendance at tutorials or other events which run weekly. Tutors maintain the records for their own tutorial group, with students restricted to only viewing their own records. Module coordinators can view and edit any records within their module(s).

TAGS also contained the following general (not subject-specific) tools for collaborative coursework:

File share Allows users to share files amongst a group. Permissions to upload/download from the file share are configured by its owner, allowing it to be used not just as a free-for-all storage space but also as staff-only file share, a drop-box for students to upload work to or a way of letting staff serve files to students.

Document Sharing Tool Similar in functionality to the file share tool, but with added functionality for students to leave comments on documents uploaded, and an interface optimised for the use of Northern College (based in Dundee).

Notebook A simple single-threaded messaging system, intended to provide a space for students to discuss team work, for example which shares to buy/sell as part of a stock trading exercise.

Questions and Answers Similar in design to the notebook, except that instead of focusing on discussions between students, it focuses on student-tutor communication. Students could submit questions to teaching staff, who could then reply either solely to the student, or share the answer with the whole class where it was relevant to a wider audience.

URL A link to some resource, typically external although links into parts of TAGS have been used before to simplify navigation.

URL Book A set of links, almost identical to the URL service.

Similar tools to many of these were developed in MMS, although the designs would be expanded significantly to handle substantially more complex use cases. Expansions to these tools included functionality such as scripted mark calculation, adding folders to the file share and discussion threads to the notebook. Finally, TAGS contained a small number of teaching tools specific to individual taught subjects. Detailed specifics of these tools are beyond the scope of this thesis, however a brief summary is provided below:

Portfolio (Finance, Accounting, Management, and Economics) Provides a virtual stock portfolio, allowing students to see what profit/loss they would have made on shares without the need for real money. Uses live (15 minute delayed) data from the London Stock Exchange. Commonly used with the Notebook resource to provide a log of why students made certain trading decisions[44][109].

Model Patients for Clinician Training (Medicine) MediCAL helped train students in diagnosis of medical conditions, using video clips of real patients presenting symptoms a variety of conditions[6]).

Ecrire/Ecrira (Modern Languages) Provides a framework for students to learn foreign languages (specifically French or Spanish) within, working as part of a group to translate a text.

Genetics Case Study Similar to the MediCAL tool, provided anonymised patient case histories for students to use in practising diagnosis.

Subject-specific tools were not ported to MMS through lack of demand; most of the projects had lapsed into inactivity, with the exception of the portfolio tool, which was not in use at St Andrews. Functionality to support such tools is however part of MMS, and the LAVA project (see 10) is an example of these.

2.4.2 INSIDE

An Institutionally Secure Integrated Data Environment (INSIDE)[13] [14][21][66][67], the second project from which MMS evolved, was a JISC funded project[56] at the Universities of Durham and St Andrews. INSIDE sought to understand the barriers to sharing student records data inside and between institutions. It provided some proof-of-concept software to show how the data maintained in legacy systems such as SITS could be made useful to the academic populace at large through web-based exposure.

The INSIDE project primarily provided insight into the difficulties inherent in using institutional information bases. In particular, it highlighted severe weaknesses in data currency, for example those caused by delays in updating the central Data Warehouse at St Andrews. One of the key tools in INSIDE provided the ability to check for and report disparities between School, Central and Data Warehouse views of students enrolled on an academic module, allowing academic staff to quickly identify such mismatches.

Lessons learnt from INSIDE meant that by the time MMS was ready for deployment steps had been taken to significantly improve data quality, and users could be forewarned of likely issues. INSIDE also provided the basis for the design of the enrolment resource type, significantly accelerating its development.

2.4.3 Finesse

Finesse, unlike the other web applications described, is not intended to help in delivery or administration of a course, but instead enables practice of concepts learnt in the course. Finesse is a stock trading simulation, where students are given a certain amount of virtual money with which to trade on the London Stock Exchange. These trades are performed using prices derived from real world current stock prices[109].

It has various use case scenarios; some institutions have students trade independently, while others have them work in groups. A shared notebook is also provided by Finesse, which some institutions have their students use to discuss and explain their strategy.

Finesse was originally implemented as a separate application, then later merged into TAGS, so it could use the same framework for authorisation and authentication. With recent changes in the requirements for Finesse, in particular ease of maintenance, moves have been made towards separating it from TAGS.

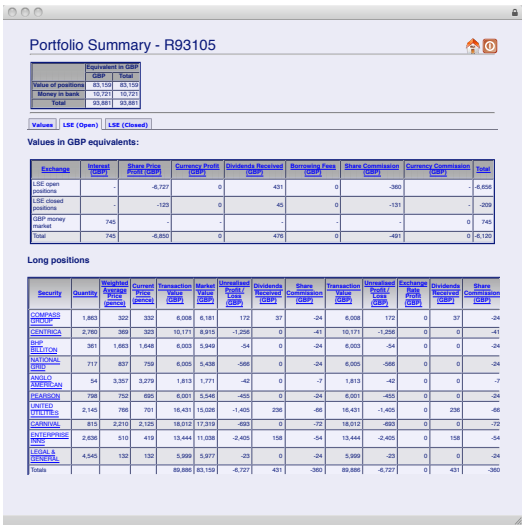


Figure 2.4: Finesse Portfolio Summary Page

2.4.4 DIF

The Disability Information Flow (DIF) project, as a sister project to MMS, also sought reduce the administrative overhead of ensuring information about students with special needs (for example lecture notes on coloured paper, or extra time in exams) was distributed to the correct members of staff in a timely manner[71].

Previously Student Support Services manually composed memos about students with special needs, which were sent to the secretaries of units responsible for teaching each student. The secretaries would then photocopy these memos, and distribute them to the lecturers and course coordinators of modules the students were taking. This process was time consuming, error prone, and created a lot of paperwork which teaching staff had to keep track of.

DIF was designed by Bin Ling and Dave Roberts[72] [4], and I was responsible for the implementation. It tracked the staff responsible for teaching each module, the administrative staff for each academic unit which modules students are taking, and the details of any special requirements for students. Student module associations and special requirements were automatically imported (although more detailed versions of requirements could be entered manually), while details of staff roles had to be manually entered due to none being maintained by central Registry.

The first version was intended as a rapid prototype, and was embedded into the MMS framework to accelerate development, although it was later split off into a separate code base. DIF was used for several years, before finally being replaced in 2007 by a functionality within SITS:Vision, which used e-Vision as a user interface for academic staff.

Lessons learnt from DIF were extremely useful in the design of MMS' API and data structures for dealing with academic units, particularly with unusual configurations such as modules being taught by staff belonging to other units. Further, the functionality was eventually re-integrated into MMS in 2010 in response to dissatisfaction with the SITS:Vision based solution.

2.5 Publishing Models

The applications described above can be split into open source (Moodle, uPortal) and closed source (ANGEL, Learn, WebCT) projects. Further, while most are generic applications which do not target specific institutions, a few (Bodington, BOSS) are in-house applications tailored more closely to the institution where they have been developed. MMS' licensing model is currently effectively closed source (copyright retained by the University of St Andrews), although permission is being sought in order to release the code.

Open source projects have the advantage for the user that they can inspect the code before committing to deploying the application, enabling them to assess in depth its suitability, and any issues arising can (in theory) be addressed by the end user without referring back to the publisher. In my experience there is often a lack of resources available to make significant changes to open source software deployed within an institution, however at least the option is available.

By comparison closed source projects are typically better resourced, with full time developer teams working on the code, however leave the customer dependent upon the publisher for updates and support. Historically closed source projects have often been better supported, although it is now common for open source projects to be available with paid support. Moodle Pty offer paid support for Moodle, and there are number of service providers for support for uPortal.

For staff at St Andrews, MMS blends open and closed source models, in that the source code is not readily available however the scripts used to implement academic policy (discussed in detail in section 7.6) are exposed to staff who are encouraged to review them to ensure accuracy of implementation.

MMS has been developed an in-house application tailored to systems and processes in use at St Andrews, which limits the application, and it could not readily be used at other institutions, although the institution abstraction layer (5.11) is intended specifically to enable such extension/replacement of functionality to support other institutions.

Chapter 3

Challenges

This chapter will expands on the design, adoption and support challenges addressed by the MMS project, focusing on the underlying causes. This work provides a basis for later chapters, which describe the solutions developed.

Wilson[137] and Yuan et al[140] examine the suitability of existing standards processes for use in a higher education context, and describe community-led informal processes which better suit the requirements of higher education software development. This provides context for higher education specific requirements and challenges.

Software design for higher education administration requires the architect to overcome a number of challenges specific to the sector, which are often overlooked, understated or/and misunderstood. Many of these have become apparent as MMS has been rolled out across the institution, both from the researchers' own experiences using the system or from user feedback. These challenges include:

- Higher education teaching across different faculties is not a uniform process[107], and as such administrative tasks in relation to teaching are not uniform either. For example, Computer Science at St Andrews typically has weekly assessed work to be submitted by students, while English may have just one or two more substantial essays to be submitted for each taught module. In cases such as Chemistry submitted work may take the form of the results of an experiment, with no digital assets submitted (this is especially true where work is recorded in lab books).

- Teaching and administration may also vary by intended degree award, such as BSc, MSc, PhD, etc. Taught degrees tend to have similar requirements, however research degrees are substantially different with cross-year deadlines, no grading process of work (instead students are typically assessed with relation to whether and how their degree should proceed). Notably these variations interact with per-subject differences, vastly increasing the number of possible process combinations. Portfolio-based degrees such as D.Eng add further complication.

- Processes also vary by attendance style; part-time, evening degree and distance learning students also have their own subtle differences in process requirements. Students, especially those attending as part of a semester-abroad programme, may have exemptions and exceptions made, such as essay-based assessment in place of final exams.

- Exceptions to normal process are extremely common, for reasons including (but not limited to) personal circumstances of a student, additional support required due to disabilities, or even to correct earlier procedural errors. Examples of exceptions include extensions to deadlines, exemption from pieces of work, changes to grades awarded and course material only made available to a subset of students on a course (such as notes in alternative formats).

- Higher education institutions can reasonably be expected to be innovative in their processes. This innovation leads to demand for support of rapid changes in requirements.

- External organisations can impose requirement changes on institutions, forcing process changes and technical requirements which are difficult to predict ahead of time. Consider the examples of introduction of attendance reporting to the UKBA for overseas students[88][127], or the suggestions of UCAS switching to admissions decision making after exam results are issued[114].

These challenges are also important to consider in terms of acceptance of new software by staff. Schneckenberg describes how lack of staff engagement can be a barrier to adoption of e-learning in higher education[116], and many of the same principles apply to adoption of software for managing administrative tasks. It is important that staff see value of new software[26] to themselves, as well as to the institution, if they are to invest effort in the adoption of that software.

3.1 Process Complexity

As a further consequence of these challenges, systems tend to be highly complex, and accordingly ensuring processes are implemented correctly is a challenge.

For software architects anticipating a more homogeneous, stable organisation, these challenges can be a cause of significant complications in the software development process, especially if they are not effectively managed at an early stage in the design.

3.1.1 School and Subject

In his 1976 work Weick illustrates the complex interactions of educational organisational elements, describing the result as "loosely coupled systems"[132]. This loose coupling means that organisational elements (such as academic schools, or taught subjects) have a significant degree of autonomy. This leads to greater flexibility in terms of introducing process differences at the school or subject level than might otherwise be expected of a single large organisation. It should also be noted that universities are large and diverse employers, with staff headcounts around the 1,000 to 10,000 range, and heavily split not just by role but also by subject.

These differences manifest in a number of ways, ranging from obvious differences such as assessment process, to almost trivial differences such as policy on when to round during calculation of final grades. It is outside the remit of one academic school to define process upon another, it is obvious that understanding and supporting these differences was a requirement for MMS, given its development within the School of Computer Science.

3.1.2 Intended Award

While there are related themes in administration of taught and research degrees, it should be obvious that such degree types do have significant differences. Taught subjects in general include a number of points during the academic year where students are assessed on what they have learnt, whereas research students generally have a more supervision-orientated process. There are clearly similarities, however intent and outcomes differ.

This form of subtle difference is typical. Other differences include processes for progression of students, such as confirming sub-honours students can progress on to honours, or M.Phil students can progress on to a PhD.

These differences interact with school and subject differences, substantially increasing the number of possible combinations. For example, undergraduate Medicine has different processes to both postgraduate Medicine and undergraduate Philosophy.

3.1.3 Mode of Attendance

The last potential complicating factor with institutional processes is mode of attendance. Distance learning, part-time, semester/year abroad and evening-degree students can all have their own requirements in relation to standard processes, especially as such degree courses can run substantially outside the normal academic year. Examples include:

- Distance learning students may require additional support, particularly in terms of feeling involved with their institution and other students, which technology is well placed to assist with[86].
- Semester abroad students may be required to complete alternative assessments in place of end of semester exams, where such exams occur after the students are expected to have returned home.
- Part-time research students generally have different assessment requirements to reflect that their progress is at a different rate.
- Part-time taught students may be taught across multiple calendar years, meaning their modules have extremely unusual requirements in terms of both availability to students, and reporting deadlines.

3.2 Acceptance By Users

It is common for decisions regarding introduction of new software systems, and changes to provisioning of software to be made by a limited set of actors within an institution[118],

leading to academics often perceiving new technology as imposed upon them. Academics may also feel that new systems and processes are designed, or at risk of, making them redundant[92]. Naturally this can lead to a reluctance towards adoption of new systems, and without the involvement of such stakeholders it is unlikely that a change will be successful[12].

MMS' adoption prior to the academic year 2010/1 was organic and academic-led, spreading from a number of modules within the School of Computer Science, to the whole school and the School of Mathematics and Statistics, with around half of academic schools opting to adopt MMS. During this early lifecycle stage MMS' adoption was dependent on academic staff perceiving MMS as assisting them. In 2010/1 all taught modules were required to use MMS for reporting, and many initially opted to limit its usage to such. Adoption has since continued in an academic-led manner, and usage statistics are discussed in depth in Chapter 11.

This adoption is primarily credited to MMS fitting academic workflows and requirements, in comparison to attempting to fit academic workflows to the system. This has required supporting a number of unusual and exceptional cases, as discussed in the next section.

3.3 Exceptions to Established Process

There are a number of exceptions to normal process, ranging from common examples such as student illness, through to more unusual cases such as modules which are joint-taught with external organisations (generally other higher education institutions). These exceptions present unusual, frequently one-off process implications.

For many of these exceptions MMS goes beyond simply enabling them, and incorporates specific functionality for managing exceptions. Examples include student exceptional circumstances (illness, bereavement, etc.) and disability tracking tools, which assist with ensuring this information is presented clearly to staff where it is relevant.

3.4 Process Innovation

While it has been shown that academics in general follow teaching approaches based on their own teaching conceptions[62], this does not preclude research by academics within their concepts of teaching, or more varied research by a relative minority of academics. This leads to a small but important set of requirement changes which are by their definition (as research areas) difficult to predict during the design phase.

These research aspects lead to exceptions to established process, which may be well outside initial design requirements. Enabling rapid changes to the implementation is therefore an important aspect of the design, beyond handling established requirements.

Further, requirement changes can be imposed by external organisations. These are typically technical or data changes, requiring changes to integrations with external systems, or provision of additional data. Data-based requirements provide the easiest examples, such as the UK Border Agency's introduction of requirements for student engagement data. These organisations generally do not have a deep understanding of the institution's systems, and may impose changes on timescales that pose a significant challenge.

3.5 Supporting a Research Project

Davis and White[25] describes issues with conflicting feedback from users, and how a software project can be a success from a research point of view while achieving limited take up, and illustrates the divergent success criteria of research and conventional software publication.

In the case of MMS, there were also difficult trade-offs in pace of research vs stability of the product. Research software is under pressure to change rapidly in response to discoveries and new theories, while software with an established user base is expected to be stable and reliable. MMS is in active use by around 9,000 students and staff, who quite reasonably expect key software to be stable and consistent in day to day usage.

The needs of research lead naturally to an agile methodology, which includes a short software release cycle, however even agile has a cycle based around weeks[63] whereas MMS' release cycle has typically been once every 4-7 days. In extreme cases requirement changes

have gone live within hours of been received, where critical functionality has not been captured at an earlier point in time.

Attempts to extend this release cycle in order to improve stability and consistency of the delivered platform have struggled with ad-hoc requirements being communicated to the researchers with a very limited delivery window. To date it has been considered preferable to meet these requirements rather than insist on a longer lead time.

Further, supporting software with a significant number of users posed further challenges. Where a conventional IT department would have specialists for tasks such as responding to helpdesk queries, managing servers, training users and administration of databases, the researchers were required to be generalists. Central IT services provided support such as provision of helpdesk software and maintenance of physical server hardware, which did substantially assist. The researchers' background in computing assisted in managing these tasks, compared to software projects led by researchers with background in other disciplines[121].

3.5.1 Effort Management

There are additional challenges when developing software intended to reduce costs, in that the costs inherent in the software must of course be less (ideally significantly less) than the costs saved through its usage. This is compounded by development within a single institution, limiting the scope for spreading expenditure. These costs can be broken down loosely into software development, staff development and user support.

Effectively managing these costs for MMS is addressed primarily by the architecture being designed to simplify development, encourage straight-forward user interface design, and simplify diagnosis of issues. Typically 2 to 3 researchers are involved in MMS at any given point in time, and communication and co-ordination issues which may arise at scale are avoided as a consequence of the small team size. Since 2009 the research budget has been funded internally by the University of St Andrews, providing stable funding for 2 researchers. Task prioritisation is typically informal, with each researcher responsible for their own work queue and passing off/requesting tasks as required to balance workload.

3.5.2 User Support and Training

Managing requirements in terms of staff development and user support are further complicated when deploying software from within a single academic school, rather than from central support units, as resources that would normally be expected to be available for such tasks are typically limited to handling institution-wide projects with backing from senior management.

Accordingly, costs for activities such as staff training, writing of end-user documentation and user support are done using the same pool of researchers. These tasks then have a direct impact on research output, meaning that minimising the needs for these is a priority at the design and development stage.

Chapter 4

Design

A central theory behind MMS is that it is possible to substantially streamline institutional administrative efficiency by using a suite of software tools to improve visibility of both data and processes. Underpinning these tools needs to be a framework providing common functionality such as authentication and authorisation, integration with external systems, custom data types, etc.

This thesis primarily discusses this core framework and its own data management tools. The design of MMS was formed around the "pain points" of academics working in the School of Computer Science, based on discussions of their experiences of existing processes. This led to an analysis of where these processes are unduly time-consuming, complex or risky.

MMS' framework provides support to researchers and developers working on tools for students and academics, with a focus on use cases involving module and/or tutorial-group centric authorisation models. The APIs it provides avoids cases of "re-inventing the wheel", and enable rapid development of tools which make use of institutional data through provision of abstracted functionality for common tasks. The core MMS tools build upon that framework and seek to minimise administrative overheads and risks through a set of complementary principles taken from experiences in developing TAGS, and refined from user feedback on MMS (both informal, and from surveys which are discussed further in chapter 11):

- Modelling roles and privileges in a manner that closely reflects actual institutional

structure[6], in contrast to designing a model based on an abstraction of the institution, and requiring users to translate roles into that model. This principle stems from experiences gained working with TAGS, and the limitations of its model.

- Acquiring data from as close to the original source as possible; for example coursework is submitted directly by students, grades by academics, attendance from tutors. Collecting data at source reduces the risk of errors during transcribing of data. This principle is self-evident when considering data flows within an institution.

- Version control stored data to reassure users both that processes have been followed correctly, and that any input mistakes can be easily caught and rectified. This principle was added after initial development of MMS, in response to user queries about the origin of data or changes to data, and the frustrations in being able to answer those questions.

- Provide detailed audit trails on data and configuration to allow users to understand how a dataset has been formed over time. This is a logical extension of version control of data.

- Exposing data as widely as practical, from the best source(s) available, while reflecting institution policy on data security and management. This principle is based on experiences of the INSIDE project, and informal user feedback about frustrations with duplicated effort in entering the same data in multiple places/systems.

- Exposing process details to end-users, to enable more meaningful analysis and feedback, by using a "many eyeballs" approach as discussed in Raymond[111]. Typically end-users are also more knowledgeable on the processes being modelled than developers, and enabling them to better understand how processes are modelled improves chances of them being able to identify issues. This principle is based on experiences supporting MMS, where staff queried models the most unambiguous description of the model was the actual code used.

- Providing tools for analysing data quality and auditing data changes. This principle originates from experiences from INSIDE, and iterative improvements to student enrolment management tools to ensure staff could answer questions arising in a self-service approach.

- Pro-actively raising issues early and clearly in order to ensure they are handled as early as possible, including identifying and flagging potential data issues for user attention. This is an extension of the common “fail fast” design in software error handling.
- Integrating with external systems such as VLEs, search appliances, web portals, mobile applications and survey tools to deliver high quality data to those systems. This follows logically from the earlier design principle of exposing data as widely as possible.

Data is exposed to the users best placed to understand it, and so incorrect or out of date data can be readily identified and corrective action taken early. MMS is also pro-active in auditing data quality, intelligently correcting issues where it can do so and contacting relevant staff where it cannot.

Naturally data security is a primary concern in this environment, and to effectively both display data as widely as possible while ensuring data is not released inappropriately MMS requires a deep understanding of authorisation within a higher education institution, and modelling the roles, privileges, entities and relationships is a critical part of its design. This also allows MMS to display the most relevant data to a user at any point, for example indicating a student’s illness records next to tutorial absences.

4.1 Lessons Learnt from TAGS

MMS’ development started in mid-2002, targeting a release in time for the start of the academic year 2002/3, three months later. This initial release was intended as a proof of concept for the design, and would provide the core application framework along with a number of tools targeting the most popular uses of TAGS. These tools included:

- coursework submission, marking and feedback tool
- external link tool
- file sharing tool

- tutorial attendance tracking tool

These tools were chosen because they were the most popular uses of TAGS. While the majority of these tools were administration-focused, this was not the reason for their selection. The administration theme did however direct later decisions in extending the platform's functionality. These tools were supported by a basic framework covering:

- New core data model intended to reduce code duplication through homogenisation of entities (users, groups, modules, resources).
- A hierarchy of groups of users for authorization modelling.
- Administrative tools to manage modules, groups, users and the relationships between them.
- System administrator tools for auditing logged in users and logging users out if needed.
- Developer diagnostic tools for checking database record integrity (MySQL had been configured to use MyISAM instead of InnoDB for the database engine, and as a result it did not provide referential integrity constraints, and at the time it was not anticipated how risky this would be).

4.1.1 Improvements in Relation to TAGS

The TAGS project provided a basic support framework for applications intended to assist with web-oriented administration and teaching, as well as hosting a number of such applications. TAGS' authorisation model was based on users, groups and resources[6]. Under this model users (who could be students, tutors and/or sysadmins) could belong to zero or more groups. Those groups owned resources (instances of tools), which were access to users in the group. Feedback from developers and users on its design highlighted a number of key limitations, which provided the motivation for MMS' development:

- The user-groups-resources authorisation model was a poor match to actual group types used in the institution. The lack of clear associations between groups in TAGS

and actual organisational elements led to groups being used to represent a number of different elements.

- The groups used were a single layer (non-hierarchical), which meant that they could not model common cases such as tutorial groups belonging to a common parent taught module.
- The data model had no concept of academic year, meaning that either material was archived into static pages (and required separate hosting), or new groups were created for each academic year.
- The roles available in the system (sysadmin, tutor and student) were too simplistic to model the complexities of actual roles in an educational institution.
- The choice of Tcl as the language TAGS was written in meant a lack of complex data types, with significant consequences both to code clarity and performance.
- The API provided had no effective risk management; for example it required that each script file called the authentication and authorisation code, and would not provide any security if this was forgotten.
- The API had evolved over time, with contributions from a number of individual developers, and consistency had suffered significantly as a result (in extreme cases, it provided multiple different frameworks for the same functionality with no clear differentiation between them!)

The users-groups-resources model was easy to understand and implement, but did not capture the complex reality of users' roles within an institution. It could not handle students who were also staff, and treated them as staff for all groups they belonged to. This was significantly problematic in cases such as postgraduate students who were tutors on some modules. There was also no representation of staff with different or multiple roles within a module. This role differentiation is important as roles infer different responsibilities and privileges, for example tutors can generally only mark work, and they cannot set assignments for students.

There was a lack of clear guidance as to what organisation element a group represented, resulting in a confusing mix of different models. In some cases a group represented a module, and was cleared and re-used each year, in others a new group was created for each

time a module ran. This was further complicated by tutorial groups; some modules might have multiple groups in TAGS, one for each tutorial group, while others might use a single group in TAGS. With no automated management of student lists, entering and updating lists of students taking each module was proving to be a time consuming and tedious task.

The TAGS application itself had a lot of very similar code for maintaining (creating, destroying, editing, and assigning) these entities. The data model itself lacked a consistent naming scheme for tables and columns. These led to both an increased risk of subtle bugs in implementation, and significant amounts of duplicate code, increasing maintenance costs and complicating further development.

4.1.2 Data Modelling

The data objects used in TAGS included a number of similar data types with relatively subtle differences in API. These typically consisted of a unique ID, a short human-readable identifier and a name, among other fields. At the time it seemed sensible to give these a common superclass, reducing code duplication (especially in interaction with the database layer).

In reality, while the general intent of the fields was a close match for the attributes being modelled, the generalisations made it trickier to understand the API and data layer. Developers who were used to modules having "module codes" could be confused to discover they now had to use the "getShortCode()" method. Similarly confusion arose over differences such as "module title" being accessed from the "getName()" method.

4.1.3 Authorisation (Users, Groups and Roles)

During the rewrite of TAGS from Tcl to Java, it became apparent that the authorisation infrastructure provided was overly simplistic, and did not satisfactorily model the real world. For example a lecturer on one module may be a group tutor on another, and require different access on each. Worse, a user may be a student for one module and staff on another, a case that was essentially impossible to handle in TAGS (this happens frequently in the case of PhD students).

TAGS' authorisation model was simplistic and rigid, requiring that users re-mapped the university's structure from a hierarchy of schools, departments, modules and tutorial groups into the one-level group model. Roles held by staff also required to be translated to one of "student", "tutor" or "sysadmin", with no finer grain breakdown of permissions.

The lack of a single definition for what a group in TAGS represented caused confusion where in some cases they represented a taught module, or a tutorial group, or a degree programme, or a whole institution.

MMS' initial draft authorisation model sought to improve the authorisation model by using a hierarchy of user groups to model the institution, for example an institution is a group (at the root of the hierarchy) which contains modules, which in turn would contain tutorial groups. Users would have then be associated at any point in this hierarchy by a relationship that also specified their role in relation to the element. A user's role(s) on one group were inherited to all groups below the one the user was associated with. A large variety of roles (including Course Co-ordinator, Lecturer, Tutor, Secretary and Director of Teaching) were defined, with roles stored in the database (rather than hard-coded as had been done in TAGS) to provide flexibility to add further roles later.

This model proved error-prone in implementation, with groups expected to represent specific elements (module, tutorial group, etc.), but without any strong typing of the data objects to make this clear. It was also extremely difficult to manage from an administrative point of view, with each level in the hierarchy managed in isolation from those above and below. Prior to launch the model was modified to have distinct data types to represent modules and tutorial groups, and later extensions added further data types to represent organisational elements such as schools/units.

4.2 MMS is not a VLE

The best known web-based tools used in education are Virtual Learning Environments (VLEs), and it is common for those introduced to MMS to initially presume it is a VLE. Applications such as Angel Learning, Blackboard, Desire2Learn, Moodle, Sakai and WebCT aim to deliver teaching content in a web-friendly format, as well as providing functionality such as online assessments (or quizzes), blogs, Wikis, etc. However, it seems these tools

start from the question “How can we improve teaching using technology?”, while MMS can be thought to ask “How can we do the same teaching, with fewer overheads?”

While there is some domain cross-over in the tools (for example MMS provides course content and quiz tools, while VLEs usually provide tools for coursework submission), MMS is focused on providing administration tools. These tools are designed to fulfil a number of goals:

- Providing a single, high quality source of data such as marks. Previously marking was primarily done using spreadsheets to store and calculate grades. These suffered from revision control issues, especially in the case of coursework or exams which are partially marked by several people. It was possible for confusion to arise over which spreadsheet was most recent, where multiple revisions existed, especially where there were multiple markers.

- Providing audit trails for changes to data, especially mark and attendance data, but also some configuration settings. This both allows for staff to undo changes, and to track who has made changes to data. As a result this improves clarity of how results have been reached and therefore confidence in the system.

- Providing trivial remote access, allowing staff to perform administrative tasks wherever they are in the world providing they have an Internet connection and a web browser. This is particularly useful for allowing external examiners to access course work, marks, etc. (and has seen some unusual uses such as delivery of digital recordings of oral assessments). It is also of significant use to staff who need to work from home or while away for conferences.

- Providing extensive archives from previous years. Coursework archives can be useful in cases such as challenges to marks given, cases of academic fraud, helping determine final degree class, etc. Content archives can be useful to track teaching materials used, especially where staff who developed content have since left the institution.

4.3 MMS is Not Solely for Academics

Virtual learning environments are primarily targeted toward academic staff, as the software is intended to improve their teaching effectiveness, with students also considered as part of their target audience. MMS was initially presented in this manner, and this was initially successful within the School of Computer Science, but as usage of MMS expanded across the university it became progressively clearer that there was a reluctance to adoption amongst the wider academic body.

In terms of improving teaching, there is a reluctance of many academics to adopt new ways of teaching[116]. White reports similar experiences[133] moving from early adopters to mainstream users.

However, with a core of administrative support tools, MMS saw significant positive feedback and adoption by secretarial and other administrative or support staff. It had originally been envisioned that modules would be set up by the course coordinators. By having setup done by module staff, the workload was expected to be spread across a large number of staff. It became apparent that in many schools secretarial staff were given responsibility for setting up modules, using details given to them separately by course coordinators (frequently copying from student handbooks).

As a consequence, administrative and management staff are also key stakeholders in MMS, in addition to students and academics, and accordingly their requirements help drive the software design. This led to some significant re-thinking of the design and usage of MMS, as well as assumptions as to who the "target audience" for such applications is. Feedback was gathered from all involved users through a number of approaches, including surveys, e-mail and in person discussion with users, and meetings of the users as a collective group. Through involving such domain experts MMS has been able to better engage with users and more effectively control risks regarding correctness of implementation.

There was discussion early on that given tools to simplify administration, academic staff may be more directly involved with the processes. This does not appear to be the case, most likely due to the already high demands on academic staff time[22]. Further, administrative staff are eager to retain involvement with academic processes, citing a need for specialist staffing and double-checking of work done, although it is likely there is also an element of self-interest.

4.4 MMS Models the Institution

Virtual learning environments are typically focused around the concept of a “course” (or module, as they’re referred to at St Andrews). Staff are assigned to either a course (with generic roles such as “Course Creator”, ”Tutor”, “Instructor”) or to the application as a whole (with roles such as “Administrator”), with typically no ability to more accurately model the actual organisational structure of the institution.

MMS in contrast has more in common with student records systems; it understands modules, groups within modules (and can model multiple distinct group types within each module, for example tutorial groups, group work groups, lab groups, etc.), and schools/units above modules (for example School of Computer Science), as well as staff who are assigned to the application as a whole (for example sysadmins). MMS currently lacks functionality to deal with a hierarchy of units (for example, it does not understand that the School of Modern Languages contains the French, German, Russian, etc. as separate departments), but support for such is intended as future work.

MMS also tries to model the roles people actually have, rather than using abstract roles. For example, where Moodle might use the role “Non-editing teacher”, MMS would use “Marker” or “Tutor”. This has proven tricky at times, as different schools do not always consider the same role as responsible for the same activities (for example, some schools have exams marked by tutors, while others restrict tutors to coursework only).

4.5 MMS Implements and Supports Policy

MMS’ core structure and tools are both designed to support implementation of academic policies. This is done both through tools which direct the user through task-appropriate workflows (such as the academic alerts and postgraduate research student administration tools), and end-user configuration and scripting tools for defining and managing policies defined at school level.

High visibility of policy implementation is core to MMS’ design, utilising a many-eyes approach to ensuring correctness. Simple scripted models are made available to staff for them to verify themselves, improving confidence in the implementation of the system while

simultaneously helping eliminate any problem-cases as early as possible. Ensuring that any issues are resolved as quickly as possible reduces the risk of complications, for example correcting a lateness model before student work has been submitted is significantly less likely to be disruptive in comparison to corrections after grades have been returned to students.

4.6 MMS Is Easy to Support

The wide variety of use cases for software in higher education institutions can readily lead to a disparate set of software to be supported, each with their own infrastructure requirements. As a consequence, support staff must be trained in use of a large number of different operating systems, databases, and other pieces of supporting software. Further, diagnostic capability of the deployed applications varies in quality and design, complicating support requirements.

MMS' framework is designed to host tools for a significant number of scenarios, as well as supporting external applications, such that it brings together a substantial "chunk" of functionality into a single common platform. This approach has advantages in terms of support, in that rather than a variety of smaller applications to manage, there is a single larger application with a well-defined, relatively constant set of supporting software.

MMS is also designed with an extensive internal testing and diagnostic suite, intended to ensure any issues are detected as early as possible, and to report these issues in a clear, consistent manner to support and/or development staff. In doing so it minimises complexity in detecting and resolving any problems, further reducing support workload in comparison to separate applications providing equivalent functionality.

4.7 MMS Supports Multiple User Interfaces

Different interfaces have different strengths and weaknesses. For example a web based interface can be used from virtually any Internet connected system, however generally requires the user take action to fetch information from remote servers (the Chrome browser has very recently added push messaging[39], however is the only browser with such func-

tionality). An e-mail interface can push messages to a user, but is relatively slow and poorly suited to sending requests to the server.

MMS accordingly provides a number of different interfaces, allowing each to be used in different scenarios.

4.7.1 Web

The web interface is the main form of interaction with MMS. The web pages are produced to follow the XHTML format, such that most web browsers report formatting errors to the end-user rather than making a "best-effort" attempt to interpret the damaged page, potentially introducing browser-specific behaviour. The page design avoids use of Javascript, in order to reduce risk of complications for users with disabilities which require specialist software (such as screen readers). Where Javascript has been used, it is strictly limited to improving an existing interface, and an alternative is available which does not require Javascript.

Following the model-view-controller pattern, it is desirable to use a domain specific language (DSL)[1] for the view, in order to simplify development of the view. DSLs whose structure loosely correlates to that of the HTML pages they produce enable rapid adoption by developers already familiar with working with HTML, and accordingly MMS' design focused on those options.

Three options were evaluated by implementation in production code; JSP[2], Apache Velocity[3] and Freemarker[4]. JSP was considered to provide a poor separation of business logic from rendering structure, as it enabled embedding of Java within the JSP document[5], and in combination with its lack of support for automatic escaping of data (see 5.9.1 for a detailed analysis of the security requirements and implications), was considered a poor fit.

Initially Velocity was trialled for a number of months, before it was decided to replace it

[1] Van Deursen, Klint and Visser[128] provide an excellent summary of the literature on DSLs

[2] `http://www.oracle.com/technetwork/java/javaee/jsp/index.html` - accessed 13th July 2013

[3] `http://velocity.apache.org/` - accessed 7th July 2013

[4] `http://freemarker.org/` - accessed 7th July 2013

[5] There was an early example during MMS' development where this was done and causing significant challenges in understanding flow of control within the page rendering

with Freemarker. Unfortunately Velocity's functionality was not well suited to the software development process for MMS. Velocity favours easier initial development by simplifying the data model and template syntax, however in doing so makes it easier for subtle errors and security issues to arise, especially in a scenario where manual testing is limited. Key problems for use with MMS includes:

- Velocity silently ignored many classes of error in the template document, such as references to undefined variables in the template context. This could lead to problems varying from subtle linguistic issues where small pieces of human-readable text were inserted by the template and could be missed in testing, to more serious problems where end-users were presented with broken/unusable pages, but developers were not notified of the problems.
- Escaping of text inserted using Velocity templates to be HTML, Javascript or otherwise encoded required explicit specification of the escaping process on every individual reference. This made it likely that human error would lead to XSS attacks either through failing to escape text, or mis-applying escapings (for example only escaping HTML entities on text to be inserted as Javascript within HTML).
- Velocity attempted to infer meaning of data types in Java for insertion into the resulting pages – this could more of a hindrance than a help in cases such as date/time objects.

Freemarker helped with these issues by:

- Providing the option of fast-fail handling of errors, such that incorrect references, missing data, or mismatched types, were more easily caught during development, and where missed until later, the automated error reporting could alert developers to the problem rather than relying on users to do so.
- Freemarker templates have control tags for defining the escaping of all text contained within the element body. This makes it easy to specify that all content within a template must be escaped as HTML, and in doing so substantially reduce the scope for errors compared to specifying encoding each time data is inserted into the template. The same can also be done for layers of escaping, such as Javascript within an HTML page, by adding additional escaping tags.

- Freemarker uses a much more explicitly defined data model. This requires substantial extra work at the development stage to define the types used, and to explicitly convert them from complex types to the simpler types Freemarker understands, however once done it provides less scope for errors and inconsistencies.

Differences are described only in so far as they relate to MMS, a more in-depth comparison of Freemarker and Velocity can be found at `http://freemarker.sourceforge.net/fmVsVel.html`.

4.7.2 E-mail Based UI

While the main interface for MMS is via web pages, it also uses e-mail for many forms of interaction with the users:

- Confirmation that an action (such as submitting coursework) has been completed successfully.
- Notification of events such as coursework deadlines, particularly in cases where deadlines have been missed.
- Reminders of tasks to be completed, such as marking of coursework.
- Alerting users and support to potential problems, for example data inconsistencies.

Independent Verification

Early implementations of the coursework tool suffered from a lack of confidence and support overhead caused by frequent student claims that they had uploaded work to the system, and it had since been "lost". Despite a lack of verified cases where this happening, this remained a persistent rumour amongst users. Staff time spent investigating such claims was also a noticeable part of support workload.

E-mail receipts were introduced to fulfil three functions:

- Provide a confirmation to the student that the work was successfully received.

- Provide proof to staff of work being uploaded.
- Provide diagnostic information for system administrators in cases where problems were reported.

Email receipts proved extremely effective in reducing the number of claims of lost files, which can be attributed both to reduction in user uncertainty with regards to whether a file was successfully received, and less mis-attributed blame for late submission of coursework. In a very small number of cases students have attempted to present fake receipts for coursework, however these can be readily verified by checking the signature on the email.

There has been only one recorded case of a receipt being issued for a file that was not stored within MMS, since the introduction of e-mail receipts in late 2002. In this instance the student uploaded two copies of the same file in quick succession (likely a double-click of the "Upload" button). MMS correctly detected that two files were being submitted for the same coursework assignment, and accepted one copy, however in deleting the duplicate copy it also deleted the original file as well. As MMS had accepted one file before this point, it had issued a receipt for the file, which was no longer present on disk.

Notification of Events

In a scenario where staff must "poll" MMS for changes and work to be completed, there is an inevitable inefficiency in terms of time spent searching for such changes and tasks, due to time spent checking unnecessarily. Using e-mail to notify staff of events, such as students uploading work, can minimise time expended checking for such activity. This is particularly important in cases where work arises to be done, outside the normal timeline, for example a student submitting work after the deadline.

In one extreme example a student was given an exception to a coursework deadline, and submitted after the end of teaching. The school noted that it required staff to actively check for this work, which is both time consuming and error-prone.

Other requests have included for automatic monitoring of student performance (attendance, average grade, timeliness) and notifying staff where these metrics are outside of a defined set of criteria. Early prototypes were developed, however they suffered from accuracy issues due to underlying data quality, and were eventually abandoned. The later expansion

in use of data in MMS has led to improved data quality, and this feature could be re-evaluated as part of future work.

Alerting Users to Problems

Timely notification of users and support staff in case of problems is key to reducing the impact of those problems. Consider a scenario where a contradiction arises in the student enrolment lists for a module, and a student is left off a module as a result. If this problem can be resolved within hours of it arising, the student may never be aware of the issue at all. As time goes on, however, the student is likely to raise a help-desk query (with consequences in terms of support time). If the problem is allowed to persist, it can have consequences in terms of access to course materials, group signup, etc. which are significantly harder to undo.

Input via E-mail

There are scenarios in which user input via e-mail may be a natural interface in comparison to a web application. This is particularly true where material is already in e-mail form, for example conversations between users, to which MMS could then be CCed in order for it to be able to track the discussion. Students have also suggested that it would be convenient to be able to email coursework to MMS, for example so that in case of network connectivity issues, retries are automatically handled by the mail infrastructure rather than requiring manual intervention.

Input over an asynchronous communication system introduces substantial complexity however, such as how to handle lost e-mails, risk of e-mails being delayed (especially where time sensitive, such as coursework submission), security implications (given that encrypted e-mail is by no means ubiquitous), etc., and remains an item for future work.

4.7.3 RSS and iCalendar Feeds

Unfortunately, it is extremely hard to provide a one size fits all solution to interaction with MMS as the users are varied in their roles, and even users with the same role frequently

have different workflows. As discussed earlier, web interfaces require action on the part of the user to pull information, while e-mail interfaces raise complex issues in terms of reliability and security.

Feeds of data from MMS were developed in RSS (Rich Site Summary) and iCalendar formats, as an attempt to find a compromise position between web and e-mail interfaces. These are both machine readable formats, intended to be consumed by news reader and calendaring software respectively. RSS is designed to allow elements of content from a site to be embedded in other sites or applications, while iCalendar (as the name suggests) provides calendaring data (such as a list of events, to-do items, alarms, etc.)

Authentication

Providing access to these data feeds introduced new security challenges – the feeds contained personal data of a potentially sensitive nature, and as such access controls were required. However MMS authentication is by submission of a web form, which was not something we could describe in a way an RSS/iCal client could interact with.

Some clients have support for HTTP authentication as part of requests for data feeds, however this is not necessarily universally supported. HTTP authentication would also been infeasible as it would require that either the client stores the user's credentials (a security risk) or prompts for them every time the client is launched (which users would be unlikely to accept).

The model adopted was to provide data feed URLs with cryptographically secure randomly generated keys included in them. These keys identified the user accessing the feed, and also acted as their authentication credentials.

RSS Feeds

The core of the data feeds was provision of a set of Rich Site Summary (RSS) feeds, from various points in the MMS application. Although RSS is also a format, it's used here as a generic term, the actual format used was Atom. Atom (as opposed to the alternative RDF or RSS formats) was chosen as it provide more functionality, especially in terms of media formats supported[93]. Data feeds were added from users' modules pages, individual mod-

ules, the coursework tool (for staff only) and the content tool (for example to list lecture notes as they're made available).

The feeds from the modules page and for individual modules were intended as navigational aids, providing lists of modules that a student is taking, or tools within a single module. These were ideal for use as smart bookmarks in a browser such as Firefox that can display an RSS feed as a drop-down bookmark. An example of this is shown in figure 4.1 – please note that this shows significantly more modules than would be typical, as the image is a screenshot of a sysadmin account.

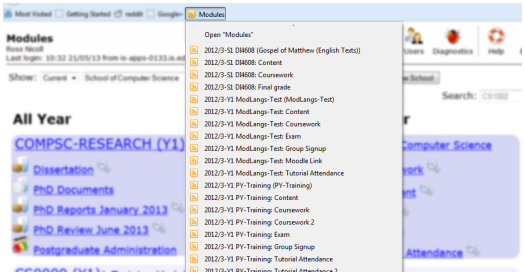


Figure 4.1: Smart Bookmarks (Firefox)

The feeds from the coursework and content tools were intended to provide a more conventional list of changes over time. In the case of coursework, the feed lists work as it's submitted by students, with the intent that staff can use it to monitor work being submitted over time. For the content tool, the feed shows a list of content as it's made available to students, for example allowing them to be notified of release of new lecture notes.

For feeds that required authentication, each feed had a randomly generated shared secret in part of the URL provided to users, which uniquely identified that user and feed combination to enable authentication.

iCalendar Feed

There was also an attempt to provide an iCalendar[6] feed of events relevant to a student – in this case details of when coursework was to be submitted (for students) or marks to be entered (for staff).

This functionality floundered as a result of a lack of good quality data. For modules where

[6]Defined in RFC5545: `http://www.ietf.org/rfc/rfc5545.txt`

submission was not done via MMS deadline information accuracy tended to be less than ideal, and there was a frustrating lack of other key data (for example lecture and tutorial times) in a sufficiently structured format in MMS.

4.8 Database Design

The data architecture was completely redesigned from TAGS to MMS to eliminate risks of data inconsistency by enforcing the third normal form[20]. The existing data in TAGS was a hybrid of several disjoint relational databases with other data stored in files on disk, and suffered from an inability to enforce foreign key constraints between databases, and race conditions relating to data stored in files.

Use of files on disk in particular for data such as notebook (discussion) entries lacked indexing, clearly defined character set encoding, and were two users to attempt to write new posts simultaneously there was no locking in place to stop one overwriting data written by the other.

MMS' data architecture eliminated use of human readable label as primary keys, which can cause issues where incorrectly set or otherwise requiring changes. Instead primary keys were generally created at recording insertion time, through the use of sequences. All data except for large files (i.e. coursework) was moved into a single large database so that constraints could be applied and transaction safety implemented.

The revised data architecture also introduced support for simple version control of data, providing audit trails as well as an ability to "roll back" mistakes made.

4.8.1 Version Control

There are a number of scenarios in which the retention of historical data is important, and system-wide support for version control of data has been critical to adoption of MMS. The ability for staff to examine changes made, as well as to undo mistakes, encourages confidence both in the system (for example by showing where a change originated) and in using the system (as users are reassured that errors can be corrected).

Consider a record of a student enrolled on a module. It is critical that there is a persistent record that a student has been on a module, even if they are no longer on the module, in order that referential integrity (with regards to coursework, attendance, etc.) is maintained, and that if they return to the module it is trivial to reverse their earlier removal. Other examples include marks awarded, coursework submitted (especially where investigating claims of plagiarism, as submitted then replaced work can be used as supporting evidence), and even configuration of tools.

As a example, to persist details of students on modules in the database, two tables are used, one to store the relationship between the student and module, one to track details of changes to the relationship. The first table contains three fields; its own primary key, plus references to the student and module. The second ("change") table has its own primary key for each record (which can be considered "revision number" for a change), a reference to a record in the first table, plus columns for the current state of the student's enrolment (active, left, auditing, withdrawn, etc.), when the change record was created, who created it, when the change was obsoleted (optional) and who obsoleted the change (optional).

Figure 4.2 shows a simplified version of this data model, with arrows representing foreign key constraints between tables. The "Module" and "User" tables on the left-hand side are intentionally left incomplete.

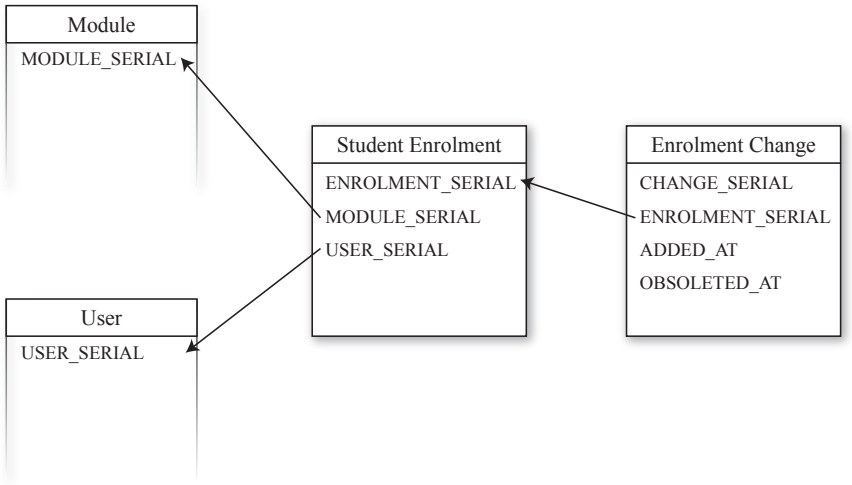


Figure 4.2: Example of Data Tables with Version Control

To enrol a student on a module, a single record is generated in both tables. To remove that enrolment, the record in the change table is marked as obsolete, but no data is actually

deleted. If the student later returns, a new record is created only in the changes table. To load details of a student's current enrolment, the first table is joined with the second, with a condition that the "obsoleted" field is null, resulting in a single record. To load all enrolments over time the same join is done without the condition on the "obsoleted" field.

For non-relationship data a similar design is implemented using a pair of tables for each version controlled entity, one to record existence of the entity and a second to store version controlled attributes of the entity.

4.8.2 File Storage Layout

Where elements of data are too considered large to be stored in the database, they are stored on risk. This data store seeks to make it easy to determine a file's path, limit number of files in a single folder, and enable human intervention to analyse the files if required (contrast with Moodle's use of SHA1 to hash file paths[7], making it essentially impossible to investigate claims of "lost"files). This manual investigation can be required for debugging purposes, or if a student claims a file was uploaded and then lost (such claims are remarkably, although only a single claim has been proven in the over a decade of MMS being used, and illustrate the need for audits and version controls).

Files on disk are stored in a hierarchy of folders. Each user tool type has its own folder under which it is free to define how files are stored, although one common pattern is used by most tool types. These top level folders are named after the tool type, using meaningful names such as "coursework", "fileshare", "exam", etc. Typically immediately below these top level folders is a folder for each individual tool which has been created, named after the primary key for the tool (a unique serial number).

Within these per-tool folders are another layer of folders, named after the primary key of the user who uploaded each file (again, this is a unique serial number). Finally, within these user folders are actually submitted files; these may have simplified versions of their original file names as part of their file name on disk, and always use the primary key of a database record relating to the file as part of the file name on disk, to avoid any risk of file name collisions. Each file on disk has a matching database record (managed by the tools,

[7] `https://docs.moodle.org/dev/index.php?title=File_API_internals&oldid=46898` – accessed 22nd February 2015

rather than in a central single table), which maps the file on disk back to its original file name. The resulting paths look like:

```
coursework/1/2/3_Document.docx
```

This approach has a number of advantages and disadvantages:

- File names are unique, meaning that an uploaded file can never accidentally overwrite an existing file.
- There are no security risks with regards to file names, as the file name on disk contains only safe characters (US-ASCII alphanumerical characters, underscore, hyphen and full-stop, although full stops are only allowed after the first character of a file name). This ensures that, if someone uploaded a file called "/etc/passwd", there is no scenario in which it might be written over the server's actual password file.
- The simplified file names on disk avoid any potential issues with file names using character sets not supported by the server file system, for example if the file name contains kanji characters. Mapping back to the original file name is handled by database records, which are stored in the UTF-8 character set.
- The use of serial numbers as folder identifiers makes it difficult to navigate the files on disk. Including elements such as the username in the folder name may be one solution, although usernames can change and as such this can cause other issues.
- The use of per-tool folders makes it infeasible for files to be shared between tools where a single file is relevant in multiple places, such as lecture notes which apply to multiple modules. This is inefficient, although a file system or storage area network which uses block dedupe to reduce duplication can mitigate this inefficiency.

4.9 Conclusion

The design seeks to accurately model the organisation and academic processes, in order to minimise user training requirements and to maximise the number of use-cases the system fulfils. MMS' design is motivated by the need to reduce costs, and as such prioritises

decisions both to reduce time spent by end-users, and in supporting the system. As a result, its functionality focuses on the administrative aspects of higher education, in contrast to many web-orientated tools which focus on the teaching and learning aspects.

Support costs are managed by preemptively assessing risks and designing components such as the data layer in order to avoid cases which are high risk, expensive to recover from, or both. This approach of preempting problems is continued through to the user interface, where MMS seeks to warn users of potential issues as quickly as possible in order to enable resolution while there is an opportunity to reduce/eliminate negative consequences.

Chapter 5

Software Architecture

Faced with the challenges identified in chapter 3, some solution was needed in order to redefine the development tasks into something which could be achieved with the resources available. To this end the following questions are considered:

- What are the common elements of functionality required for the tools? Of these common elements, which can share code?
- What design patterns are of relevance to this framework? Additionally, where common design patterns are a poor fit, can they advise new work?
- What dependencies exist between tools and how can these be managed such that developers can work in parallel with minimal interaction required between developers working on separate tools?
- How can language features be used to reduce risk of errors?

Suitable common elements in authentication/authorisation, organisational structure inspection and management, error handling and reporting, and security policy enforcement were identified. MVC and REST designs form the basis of much of MMS' rendering architecture. Appropriate isolation of developers is provided through the tool data source interfaces, ensuring those working with tools only need to be aware of the core framework. Leveraging of language features focuses on use of highly detailed types and type safety to minimise risk of accidental mixing of data types even where there is a common underlying type (for

example student IDs and staff IDs are incompatible types, although both are string-like data types).

The resulting framework empowers developers, and it would have been infeasible to produce and maintain the tools without its support. Some elements of this architecture (MVC pattern support, RESTful services, content rendering) are now available as common libraries, however the majority of it is highly tailored to higher education.

The institution model in particular is much more closely mapped to the actual institution and its data structures than is common for other web tools, and drives the authentication and authorisation support which is leveraged to allow accelerated development of tools.

5.1 Overview

The architecture is designed with a number of goals in mind:

- Consistent input processing: Handling of requests from the user should be consistent and clear, following the rule of least surprise[1][112]: Accordingly, the architecture encourages and simplifies tool/user interaction in a consistent and predictable manner.
- Abstraction of the organisational structure: Tools do not require a detailed understanding of the institution or its structure, in order to interact with institutional data.
- Rapid diagnosis: Developers are notified by e-mail of any serious problems, and detailed reports used to assist them in locating the cause.
- Security: Enforcing security through the software architecture wherever possible, for example requests go through a standard validation process before they are passed to the embedded tools.
- Separation of concerns[33]: Tools are isolated from each other, such that changes to individual tools do not affect the system as a whole. Abstract interfaces defined in the core framework provide views for tools to implement where they wish to expose data for re-use elsewhere in the system.

[1] Also referred to as “Principle of least astonishment”.

Confidence in the system has been critical to adoption, as for the majority of its lifecycle it has been adopted on a voluntary basis by academic schools. A high profile failure such as the one at the University of Southampton[43] could have irrecoverably damaged MMS' reputation, prematurely halting its adoption. To manage this, a number of different design elements were used:

- Defensive – The APIs provided by MMS' core are designed such that bugs are exposed as early as possible in the development cycle (ideally at compile time or during design).
- Type safety – Language type safety is utilised wherever possibly by encapsulating incompatible data (such as grades on different scales) within data types the compiler will refuse to interchange, even where the contained data type may be compatible (in this case, an integer value).
- Integrated tests – MMS includes its own test infrastructure for executing integrated tests on hosted tools.
- Data confidence – MMS has been designed such that neither it nor its hosted tools actually delete stored data, instead marking obsolete records as such in the database. This ensures confidence that MMS will not (and arguably cannot) lose data.

A high-level overview of MMS' architecture and interoperation with external systems is shown in figure 5.1. Here lines reflect connectivity between components and systems, and arrows show direction of data flow. Components within the central box are part of MMS itself. Of specific note are the authorisation and routing layer (referred to as "dispatchers") between the user and the tools, and the institution abstraction layer which manages data transfer to/from external data sources.

5.2 Technology

MMS has been developed in Java using Apache Tomcat to provide servlet support. The relational database used was originally MySQL with a move to Oracle early in the project lifecycle. Java was used as it was seen as a good compromise of speed and ease to develop,

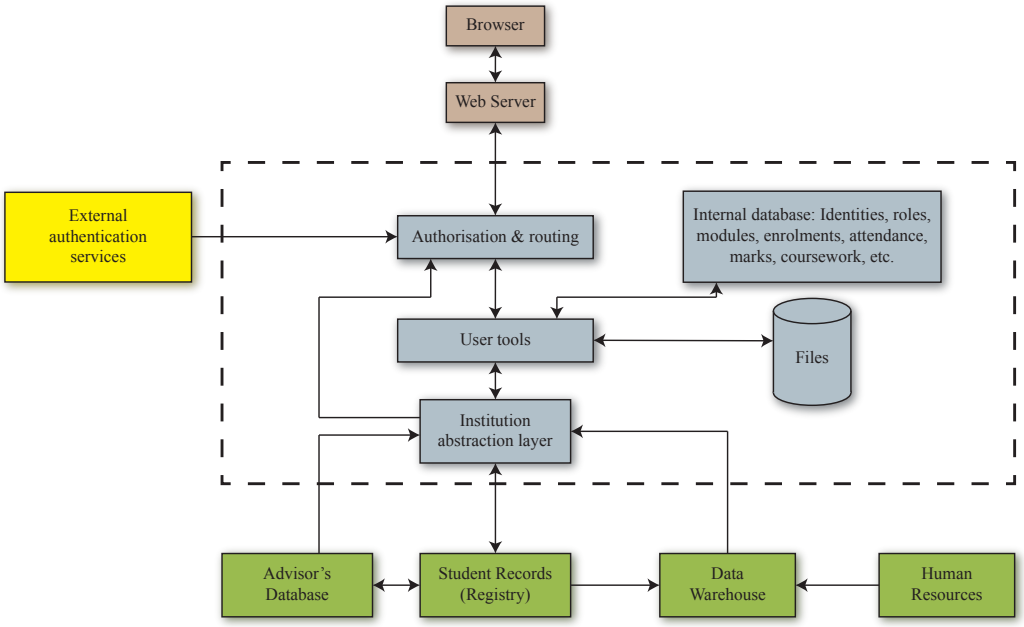


Figure 5.1: MMS Architecture Overview

with type safety and complex data type support, both of which are lacked by Tcl as used for the TAGS project.

5.3 MVC Pattern

When designing the structure of a web application, the MVC (Model, View, Controller) design pattern is widely considered an obvious choice. This can be seen by the number of web application frameworks which make use of the pattern, such as Ruby on Rails[2], Yii[3], Spring[4] and CompoundJS[5]. Designing software following the MVC pattern provides several advantages to the application:

- Improves code readability due to task isolation, so rather than mixing code to load and save data in with HTML rendering, each task is performed separately.

[2] `http://guides.rubyonrails.org/getting_started.html` - accessed 6th July 2013

[3] `http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc` - accessed 6th July 2013

[4] `http://static.springsource.org/spring/docs/3.2.x/spring-framework-reference/html/mvc.html` - accessed 6th July 2013

[5] `http://compoundjs.com/docs.html` - accessed 30th June 2013

- Changes to the data model only need to be made in one place, rather than in each different view.
- Different views of a model can be provided relatively easily, as they can share common code for accessing the data model. These views can include simple cases such as displaying different attributes of a model, or substantially different output formats (such as PDF, JSON, etc.)

The original TAGS software did not cleanly separate the model, view and controller, causing the code to be overly fragile. In many cases there were no data classes to represent the model, and it was common for scripts to intermingle data retrieval, update and rendering code. Adopting the MVC pattern was one of the major changes in the design of TAGS 2.0, although that software did not initially use templates for page rendering, which would normally be expected for the MVC pattern.

The MVC pattern was initially designed with around use cases where rendering was performed on the same physical machine as the data (model) was held[65], and in the case of client-server usage such as web applications, it is conventional to treat the client-side processing and rendering to be outwith the scope of the server-side software design. The original pattern also specifies that only a single controller is to be active at a time - taken literally this is clearly impractical in a multiuser system, especially as users can have multiple web pages open at a single time.

There is therefore some scope for interpretation in adapting the MVC pattern to the web. Leff[68], Guangchun[74] and Selfa[117] all describe solutions to adopting MVC for the web. In common with these, and in comparison to the original MVC model, view rendering is driven by the controller, rather than the model; this is required as many views can be in use concurrently, and the controllers are best placed to understand which view is most appropriate for each request.

More significantly MMS' interpretation of MVC adds an extra layer between the user and the controller, referred to as "dispatchers". This is done to better support implementations of higher education administrative tools. This dispatcher layer encapsulates common controller behaviour such as authentication, authorisation, request routing, template rendering and error handling. These act primarily as gatekeepers to the system, and validate requests before they're passed to controllers provided by views.

There are similarities to Guangchun's work in that both approaches split the controller element however motivation and method are both significantly different. Guangchun splits the controller into three (rather than two) parts, with the intent of decoupling elements to improve flexibility, whereas MMS splits the controller to move shared functionality into a common element of code.

Dispatchers fulfil a similar function to the "StandardSystemController" described by Krasner and Pope[65], although with substantial differences (for example authorisation handling). In contrast to Presentation-Abstraction-Controller (PAC)[23], which also adds additional layers to the MVC pattern, MMS' design adds a distinct layer which only interacts with the user and controllers, rather than creating a hierarchy of MVC layers.

Architecturally there are similarities to use of aspect oriented programming (AoP) to place code around existing methods, and moving to an AoP based solution might be one option for increasing flexibility. However doing so would detach security and validation constraints from the underlying code, potentially complicating modelling the flow of the code (it may potentially be unclear from a simplistic examination of the code, that other code is wrapped around it at runtime).

The details of the dispatchers are expanded upon in the later sections of this chapter.

5.4 RESTful

In his 2000 thesis Fielding describes the Representational State Transfer (REST) architectural style. This includes a number of constraints which a system must adhere to, in order to comply with the REST style[29]. While intended primarily for web service APIs, the constraints are of general relevance to web applications. Complying with these constraints infers similar benefits to web applications as they do to web services, especially in terms of compatibility with other layers in web transport (such as caches, proxies, etc.)

Adoption of the RESTful style is in accordance to the design principles of exposing process details and integrating with external systems.

Of the constraints specified, MMS complies with the majority, including client-server, stateless, cache (content is served with an indication of whether it can be cached, and for

how long), layered system (the server makes no assumption about proxies, load balancers or other layers between it and the client) and code-on-demand.

The remaining constraint, uniform interface, has proven to be challenging due to a combination of early design decisions, and limitations of request methods supported by HTML forms.

5.4.1 URLs

The URL addresses used in MMS were initially designed to follow a model similar to TAGS', where the path element represented a Tcl script on disk, for example:

```
https://.../mms/coursework/Student?module=17&user=23
```

This was done to simplify the adoption process by staff used to the previous design. However the decision to use existing design rather than re-evaluating the design at the time has complicated attempts to move to URLs which are more suitable for the Uniform Interface pattern.

URL design in MMS is described in detail in the next section of this chapter.

5.5 URL Design

URL structure within MMS specified in accordance with the design principal of exposing implementation detail. The URLs use natural keys for entities, and are structured such that they match the hierarchy of entities they relate to. Natural keys mean it is clear what the URLs correspond to, as well as improving discoverability of URLs. Users who see an address containing a module code, are likely to understand they can replace that module code with another and arrive at the referenced module. The more common case of using primary keys in URLs is often opaque as the keys have no meaning to the end user.

In having URLs model the hierarchical structure, the system enables the dispatchers to handle many security constraints, minimising code duplication and in doing so reducing risk of subtle errors in implementation. This ties in with the design principal of modelling

the institution. Security implications of URLs are discussed in depth in the authorisation section of this chapter, see 5.7.2.

For example a URL referring to the list of students taking the module CS1002 in semester one of the academic year 2012/3 might be represented as:

```
https://example.edu/mms/module/2012_3/S1/CS1002/Students
```

The path up to and including "/mms/module/" is consumed by the web server, which maps the path element to the module dispatcher, and passes control to it. The module dispatcher then consumes the elements "2012_3/S1/CS1002/", which identifies the module being accessed, and "Students" which identifies the controller to pass the request to.

This structure means that a user who knows a module's code and when it was running would trivially guess where the module would be in MMS' address space, as well as making it easy to determine where a URL referred to (such as when presented in printed material).

For tools instances, addresses identify not only the tool, but also the module they belong to, for example:

```
/mms/module/2012_3/S1/CS1002/Coursework/Student
```

Figure 5.2 shows how a URL is broken down into different elements which are mapped to a dispatcher, module, tool and finally entities within that tool. Each coloured layer represents a different component of MMS, the solid arrows progressing from top-left to bottom-right represent the request from a user traversing those layers. Text in italics describe the condition to be applied to the request before it can proceed to the next layer (dashed arrows are used to indicate where a request can be rejected).

Requests for tools pass through the same code-path as requests for modules, before the tool path component is parsed and the request passed to a controller within the tool. This is important to the authorisation model, and will be discussed further in section 5.7.

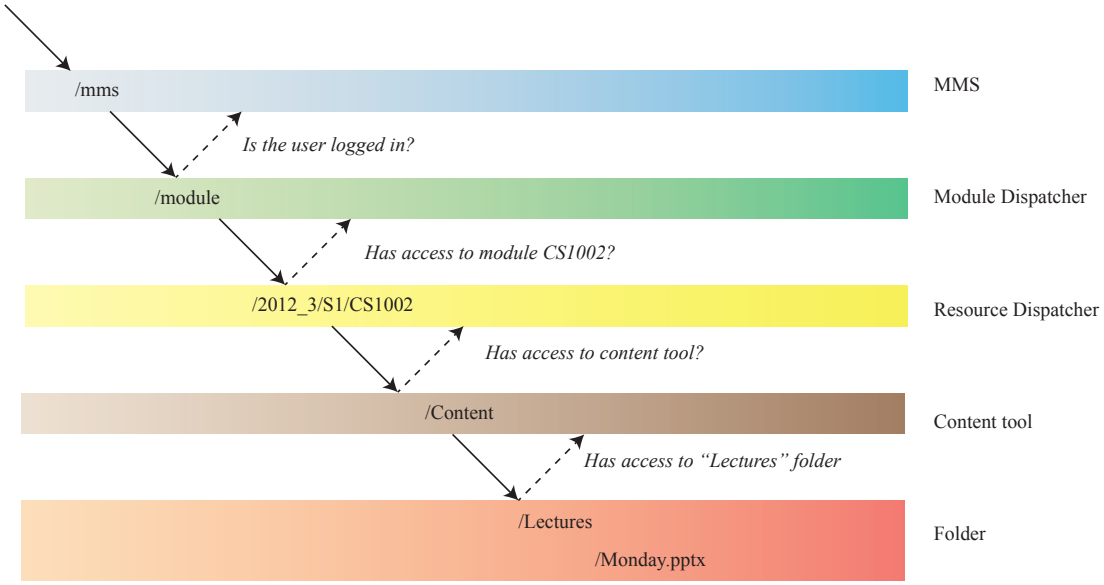


Figure 5.2: Decomposition of URL

5.5.1 Natural Keys for Content

With extension of the content tool in preparation for migration of content from WebCT, there was a need to support relative addresses, so that HTML pages uploaded to the tool can access images, stylesheets and other content at meaningful addresses. The tool dispatcher was modified to pass any remaining URL elements to the tool to be resolved into entities. For example an address for a file, before and after rewriting to the new structure, might appear as follows:

```
/mms/module/2012_3/S1/CS1002/Content/File?serial=3
/mms/module/2012_3/S1/CS1002/Content/index.html
```

Where alternative views of a file are required (for example to edit settings on a file), these are specified through use of a query parameter, for example:

```
/mms/module/2012_3/S1/CS1002/Content/index.html?view=edit
```

5.5.2 Natural Keys for Coursework, Exams, etc.

This model can be extended to cover many other types of data, improving both readability and discoverability. Coursework submissions can be modelled in terms of assignment and student, and revisions specified if needed. For example:

```
/mms/module/2012_3/S1/CS1002/Coursework/Week1/
/mms/module/2012_3/S1/CS1002/Coursework/Week1/970123456/
/mms/.../Coursework/Week1/970123456/?view=download&revision=1
/mms/module/2012_3/S1/CS1002/Coursework/Week1/?view=spreadsheet
/mms/module/2012_3/S1/CS1002/Coursework/*/970123456/
```

These URLs would reflect, in order:

1. All coursework submitted for the assignment "Week1" of CS1002.
2. Coursework for student 970123456 to the assignment "Week1" of CS1002. This would generally be the view showing details such as mark, feedback, mark history, etc.
3. Download the first file submitted by student 970123456 to the assignment "Week1" of CS1002.
4. A spreadsheet of marks for coursework submitted for the assignment "Week1" of CS1002.
5. All assignments for the student 970123456.

Exam URLs could also be structured to end in entity references of the form "<student>['/' <section> ['/' <question>]] to provide similar benefits. Constraints on development time and complexity of managing large scale changes to addressing schemes have meant these have not been adopted in actual code, however the theory should be clear from these examples.

5.6 Authentication and Identity Management

Authentication within MMS is generally dealt with by the dispatchers before a request is passed to a controller. Users are authenticated by an institutional single-sign-on service, and their identity passed to MMS by the Tomcat server. Notable exceptions to this include the web service dispatcher; again this is for legacy reasons rather than any technical limitation.

The dispatchers look up the user within the MMS database, and where an active user is found, the request then proceeds on to the authorisation step. Actual implementation is by a common superclass of dispatchers which require this functionality, meaning code to perform this check has to be written (and tested) only once. As the authentication process is a prerequisite of the authorisation process, which itself is required before a request is passed on to the controller, controllers are never called without a valid user.

This removes risk of a controller's code failing to check for a logged in user, or failing to verify that the user is active. This risk of course still exists in the shared dispatcher code, however as it is only implemented in one place, it is much easier to test and verify this code.

5.6.1 User Identity

User identities within MMS are managed by the institution's central identity management (IDM) system. This system creates users directly in MMS' database as needed, and as it is event driven (from systems such as student records, admissions, etc.), there is a minimal delay between a user account being created centrally, and becoming available in MMS. The use of IDM is in accordance with the design principle of acquiring data from as close to the original source as possible – while IDM records are created from other sources, it is the best source of data as it resolves any discrepancies between sources (for example where someone is both student and staff).

As with other data stored in MMS, user records are never deleted from the MMS database but instead are flagged as inactive if their account is no longer current. This has benefits in maintaining data integrity (for example, retaining details of students who have submitted work, after they have graduated), controlling risks (such as accidental deletion of accounts)

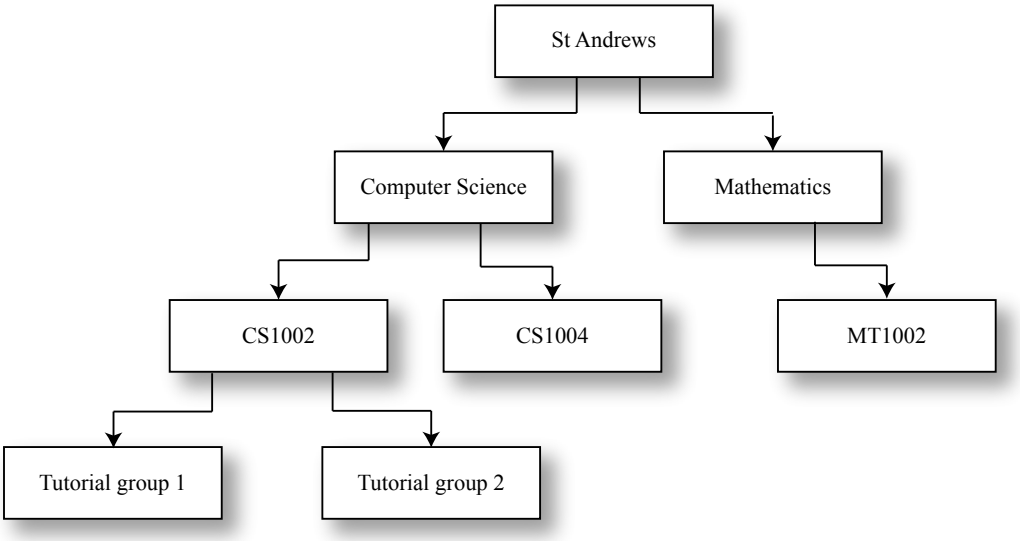


Figure 5.3: MMS Authorisation Model (Prototype)

and simplifying user state changes (for example, if a student who graduates then returns later their account can be trivially returned to the active state, rather than a new account being required).

5.7 Authorisation

Authorisation within MMS follows (and is an exemplar of) the design principle of closely modelling the institution. The data used for authorisation is of relevance to other systems, and it further embeds the principles of integrating with external systems and exposing data as widely as possible. Other design principles apply (auditing, version control, data analysis) to the supporting tools and will be discussed in section 7.2.

Authorisation is based around roles and privileges. Roles are assigned to users, based on a relationship to one or more organisational entities such as faculties, academic schools, modules, or tutorial groups.

The original design MMS' authorisation design had been to model the institution as a hierarchy of homogeneous groups, with no predefined height or structure. In this model the group at the root of the hierarchy would typically represent an institution, containing academic schools below it, which in turn would contain modules, and tutorial groups below

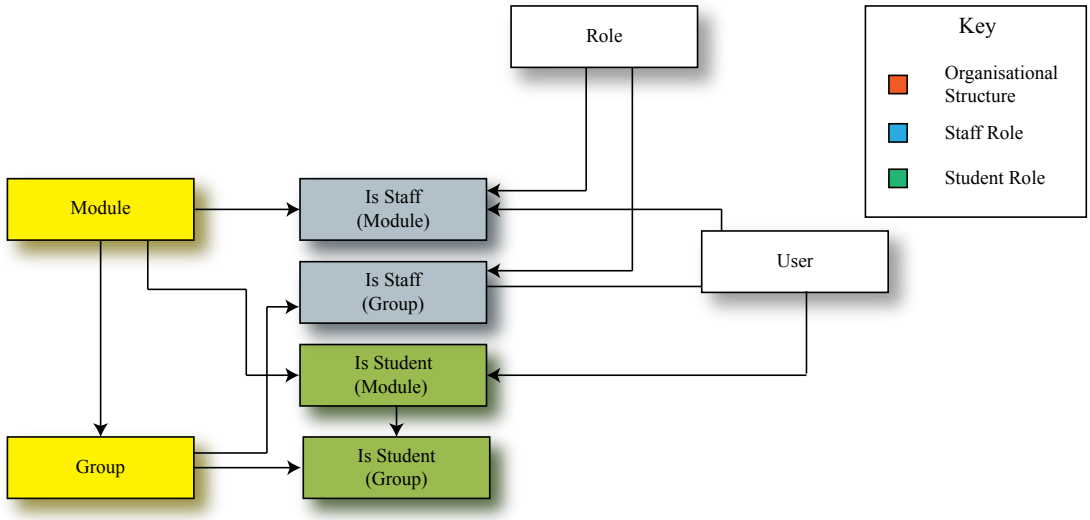


Figure 5.4: MMS Authorisation Model (2002)

them. This model can be seen in figure 5.3.

Users would be associated with groups in a many to many relationship, with roles attached to each relationship. However users were confused by the lack of structure – they expected modules to be called modules, and tutorial groups to be called tutorial groups, and trying to refer to everything generically as a "group" caused uncertainty over usage. This was both a problem for the staff entering these details, and for students trying to navigate the resulting mixed setup.

When MMS went live in 2002/3 it was limited to modelling modules and tutorial groups immediately below them. This was significantly simpler both to implement and for users to understand. This model is shown in figure 5.4.

Key changes to the core authorisation model, compared to TAGS, include:

- Core data types now shared a common superclass, with the intent of reducing code duplication in managing database records for common fields (serial number, name, description, etc.)
- The authorisation process was split in two separate code paths, one for students, one for staff, in an attempt to reduce risk of security issues relating to roles with incorrect privileges, privilege escalation, incorrect tests for specific privileges, etc. The intent was that this would avoid any risk of staff privileges being assigned to student roles,

as the two were fundamentally incompatible.

- The number of roles used for authorisation was vastly expanded, from 3 (student, tutor, sysadmin) in TAGS to over 90 in MMS. The initial set of roles was developed based on analysis of school-level roles performed as part of the INSIDE project, however roles have since been expanded to encompass support units, faculties, and institutional roles (such as Vice Principal). The very large number of roles is in part due to variation of roles by school or unit.

- The relationship between a user and a module or tutorial group specified a role for that relationship, whereas TAGS had associated roles with a user account as a whole. This change allowed accurate modelling of cases such as a postgraduate student who is a student on one module, and a tutor on an undergraduate module.

- Roles were inherited from modules down to tutorial groups; a lecturer on a module was automatically a lecturer for all tutorial groups within that module.

Later revisions would add functionality including support units with institution-wide access, academic schools, anti-roles and per-school role definitions. By 2011 the model supported a significant variety of organisational entities, which were arranged in a hierarchy. From the root of the hierarchy down, these entities are:

Institution Special case for senior management, sysadmins and other singular roles

Shard Support units such as Library or Registry, who act across the whole institution.

Faculty Collection of academic schools (i.e. Arts, Science, Divinity or Medicine)

Unit Any other units and schools

Module Taught module

Group Group (tutorial, lab, etc.) of students within a module

Staff can be assigned to any of these points within the institution, using one or more roles, while students are limited to being assigned to modules and groups (students also require to be assigned to a module before they be added to groups within it). Privileges inferred

from roles are aggregated together where a user has multiple roles on the same entity, with the exception of the special handling of students detailed below.

Students and sysadmins are handled as special cases. The student role is what has been termed an "anti-role" to staff roles, which means that assigning a user as a student to a module negates any staff role the user has in relation to that module, even the "sysadmin" superuser role.

The sysadmin role bypasses all privilege checks except the student anti-role rule, meaning they can always access any part of MMS. This simplifies developer work on new privileges (as sysadmins do not require to have privileges explicitly assigned to them), as well as acting as a safety net in case the database is corrupted.

Academic schools can define their own roles to be used for staff assigned to the school, as well as modules and groups underneath the school. Default roles are provided for the common cases; Lecturer, Course Co-ordinator, Tutor, Secretary, Sysadmin, Director of Teaching, etc. School role definitions can have different privileges to the default, for example a school that wishes to restrict entering of marks to secretarial staff can revoke that privilege from their lecturers, course coordinators and tutors.

Figure 5.5 illustrates the associations between users, roles, and organisational entities. Here lines are used to indicate a relationship between records, arrows to indicate one to many relationships (such as a module containing many group types, which in turn may each contain many groups). Clearly this model is substantially more complex than the initial design – it is (as would be expected) significantly more powerful as well, however this complexity requires to be managed carefully.

5.7.1 Authorisation API

The MMS architecture seeks to both make authorisation as straightforward as possible to use, and minimise scope for errors during development. To this end, the dispatchers act as a first line of defence, performing security checks before each request is passed to the controller. This limits the number of places in code where security checks are required, moving the complexity and risk away from development of individual pages and into shared, well tested framework code.

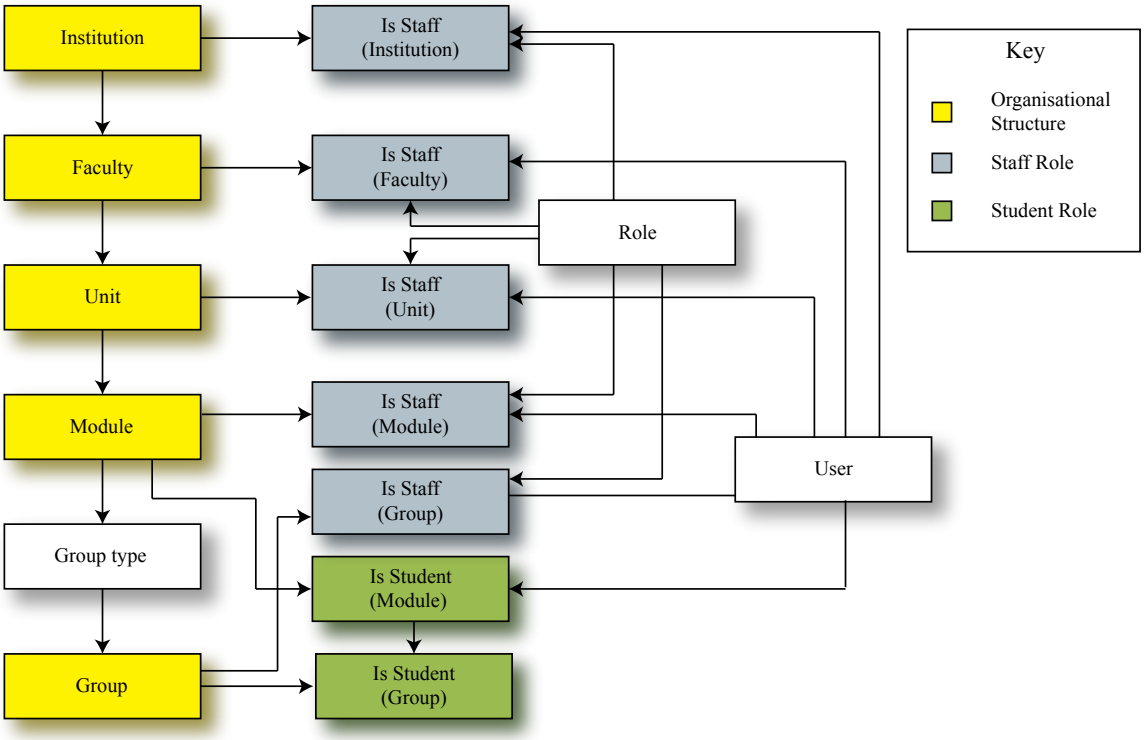


Figure 5.5: MMS Authorisation Model

There are individual dispatchers for each authorisation entity types within the institution, with the exception of groups. Each dispatcher has an interface which controllers are required to implement in order to be compatible with the dispatcher. These interfaces have a number of common elements; they define methods for handling HTTP GET and POST requests, as well as methods for retrieving the access privileges required for requests of each type. The request handling methods are specific to each dispatcher because the method signatures include the relevant entity type. For example a module-level controller implements methods which require the module being accessed as a parameter. An extract from the module controller interface is shown in figure 5.6 as an example of this design implemented in Java.

The methods to retrieve the privileges required for accessing the page are the key here – by defining them as methods which controllers have to implement (as opposed to a model such as Spring's, where authorisation details are optional annotations or XML configuration), the design mitigates the risk of permission details being missed during implementation. Further, as the dispatcher handles these permission checks, there is minimal risk of a controller failing to implement the checks correctly.

```
public EnumSet<Privilege> getActionPrivileges();

public EnumSet<Privilege> getRenderPrivileges();

public Template doAction(HttpServletRequest request, HttpServletResponse response,
        Map<String,Object> root, Institution institution,
        User user, Module module)
    throws HTTPThrowable, IOException, MMSException, ServletException, SQLException;

public Template doRender(HttpServletRequest req, HttpServletResponse resp,
        Map<String,Object> root, Institution institution,
        User user, Module module)
    throws HTTPThrowable, IOException, MMSException, ServletException, SQLException;
```

Figure 5.6: Extract from Module Controller Interface

Note also the presence of a user object (representing the currently logged in user) as a parameter to both of the doAction() and doRender() methods. This is included in the parameters to emphasise the contract provided by the dispatcher that before the dispatcher will pass a request to the controller it verifies that there is a logged in user who has access to the applicable entity Accordingly a developer working on a controller can be confident of three key points whenever these methods are called:

- That there is a currently logged in user.
- That the entity specified in the request URL is valid.
- That the currently logged in user has the privileges specified by the controller, on the entity specified in the URL.

5.7.2 Path-Based Authorisation

URLs in MMS are designed such that they include authorisation elements in their path. This enables multipart authorisation to be done in the dispatcher, as it evaluates each path element in turn. Moving on from the simple example of a module URL in 5.4.1, a tool contained within a module might have a URL such as:

```
https://.../mms/module/2012_3/S1/CS1002/Coursework/
```

Here the path elements specifying a module must be evaluated by the dispatcher before the tool identifier (“Coursework”) is evaluated. Accordingly, security checks for module access are completed before the tool is resolved. Further security checks may then be performed at the tool level, such as whether a student is resitting the module if the tool is for resitting students, before the request is passed onto the controller (in this case identified by “/”).

This shared code design then continues through the URL evaluation; the code for verifying authorisation against a tool knows that the module checks must have passed for it to be called, and likewise the content tool knows that checks must pass before it receives details of the specific file being accessed.

Content Tool

The content tool extends this model by introducing permission checks on each item of content, and accordingly on each path element. These permission checks on content are done to restrict visibility of individual items from students (either based on time, such as release of lecture material, or permanently in the case of material intended only for staff). Consider the URL:

```
/mms/module/2012_3/S1/FR1001/Content/Content/Grammar%20notes/?view=icon
```

As before there is a module identifier and tool identifier (the first line), which are consumed by the module dispatcher. The module dispatcher then passes the remainder of the URL (“Course%20content/Grammar%20notes/?view=icon”) on to a dispatcher provided by the content tool. This tool-specific dispatcher then evaluates the remaining path elements, resolving them into the content folders “Course content” and “Grammar notes”. This resolution process also verifies access at each step, meaning that were the user to not have access to “Course content”, it would abort at that stage before processing the next path element.

The content tool dispatcher passes the request on to the controller only if each path element exists and the user has access (the controller to use is identifier here by the query “view=icon”). In this way the controllers neither have to understand or implement the security model, as they are never passed requests from users that do not have the relevant authorisation. Security checks are implemented exactly once, in the content tool’s

dispatcher, and then apply universally to all of its controllers, with no risk of controllers failing to check permissions.

Future work could include expanding URL structure to cover tool-level elements such as coursework assignments, exam sections, etc. to enable the shared authorisation model to be used in those cases as well.

5.7.3 Limitations

There are some complex cases where controllers are required to do further security checks after a request has been passed to the doRender() or doAction() methods. These cases arise, for example, where a controller may show or hide specific items of data based on a user's privileges (such as in the case of anonymised marking). These are however exceptional, and almost universally the default privilege checks are still used to enforce coarse grain controls on requests.

5.8 Error Handling

One of the most critical, but frequently overlooked parts of the MMS framework is the handling of server-side errors. This is the main exemplar for the design principle of pro-actively raising issues both early and clearly, and is a keystone to minimising support overheads for MMS.

As discussed in section 5.3, there are common code elements referred to as "dispatchers" that handle common elements in processing virtually all requests made to MMS. These dispatchers catch all exceptions (in Java terms, they specifically catch subclasses of java.lang.Exception, but do not catch subclasses of java.lang.Error) thrown by the controllers they delegate to, and provide a common set of handling for errors raised:

- Log the exception to a file stored on the server.
- E-mail the support team (who in this case are the researchers) with details of the request made, the current user, the exception and a complete stack trace of the exception.

- Present a relevant error page to the user.

The use of common dispatchers to handle most requests ensures that this error handling is consistent throughout MMS.

5.8.1 Error Report Emails

Notification of support by e-mail in case of server errors has enabled rapid elimination of issues as they arise, without waiting for users to report the issue or for the error to be spotted in a log file. The e-mail messages are designed to contain sufficient information to reproduce the error on a development server, and the inclusion of a stack trace as part of the error details means problems can frequently be diagnosed from the error report e-mail alone. A sample email is shown in figure 5.7.

Rapid direct notification of the researchers in case of problems has been key to confidence in the system, given the rapid release schedule, as it minimises time to fix problems and therefore visibility of issues.

5.8.2 Error Pages

Customised error pages are provided for most client-side (400 series) and server-side (500 series) HTTP errors, such as 404 (Not Found), 403 (Forbidden) and 500 (Internal Server Error)[28]. A sample error page for a 404 error is shown in figure 5.8. These customised error pages serve three purposes:

- Ensure a coherent look and feel throughout the application, while also being visually different from other pages to ensure it is clear that an error has occurred.
- Provide appropriate guidance on how to proceed, for example checking the address, contacting teaching staff and/or the service desk, etc.
- Hide internal details of MMS from the end-user (default servlet error pages include stack traces for debugging).

```
Server-side error dealing with request for "/mms-jrn/user/me/Modules".

Status code: 500
Error message: java.lang.Boolean cannot be cast to java.util.Date
Current user: jrn
Client hostname: 138.251.194.184
Client browser: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.151
Referer: https://distsyst-dev.cs.st-andrews.ac.uk/mms-jrn/user/me/Modules
Server hostname: distsyst-dev.cs.st-andrews.ac.uk
Parameters:
unit: 1114

academic_year:

command: Get All Modules

Caught exception: java.lang.ClassCastException

Stack trace:
java.lang.ClassCastException: java.lang.Boolean cannot be cast to java.util.Date
at org.hibernate.type.descriptor.java.JdbcTimestampTypeDescriptor.areEqual(JdbcTimestampTypeDescriptor.java:41)
at org.hibernate.type.AbstractStandardBasicType.isEqual(AbstractStandardBasicType.java:196)
at org.hibernate.type.AbstractStandardBasicType.isSame(AbstractStandardBasicType.java:186)
at org.hibernate.type.AbstractStandardBasicType.isDirty(AbstractStandardBasicType.java:222)
at org.hibernate.type.AbstractStandardBasicType.isDirty(AbstractStandardBasicType.java:218)
at org.hibernate.type.TypeHelper.findDirty(TypeHelper.java:294)
at org.hibernate.persister.entity.AbstractEntityPersister.findDirty(AbstractEntityPersister.java:3820)
at org.hibernate.event.internal.DefaultFlushEntityEventListener.dirtyCheck(DefaultFlushEntityEventListener.java:536)
at org.hibernate.event.internal.DefaultFlushEntityEventListener.isUpdateNecessary(DefaultFlushEntityEventListener.java:243)
at org.hibernate.event.internal.DefaultFlushEntityEventListener.onFlushEntity(DefaultFlushEntityEventListener.java:172)
at org.hibernate.event.internal.AbstractFlushingEventListener.flushEntities(AbstractFlushingEventListener.java:225)
at org.hibernate.event.internal.AbstractFlushingEventListener.flushEverythingToExecutions(AbstractFlushingEventListener.java:99)
at org.hibernate.event.internal.DefaultFlushEventListener.onFlush(DefaultFlushEventListener.java:51)
at org.hibernate.internal.SessionImpl.flush(SessionImpl.java:1127)
at uk.ac.stand.mms.institution.Institution.close(Institution.java:182)
at uk.ac.stand.mms.servlets.AbstractDispatcher.service(AbstractDispatcher.java:319)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:722)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:305)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
```

Figure 5.7: Sample Error Email

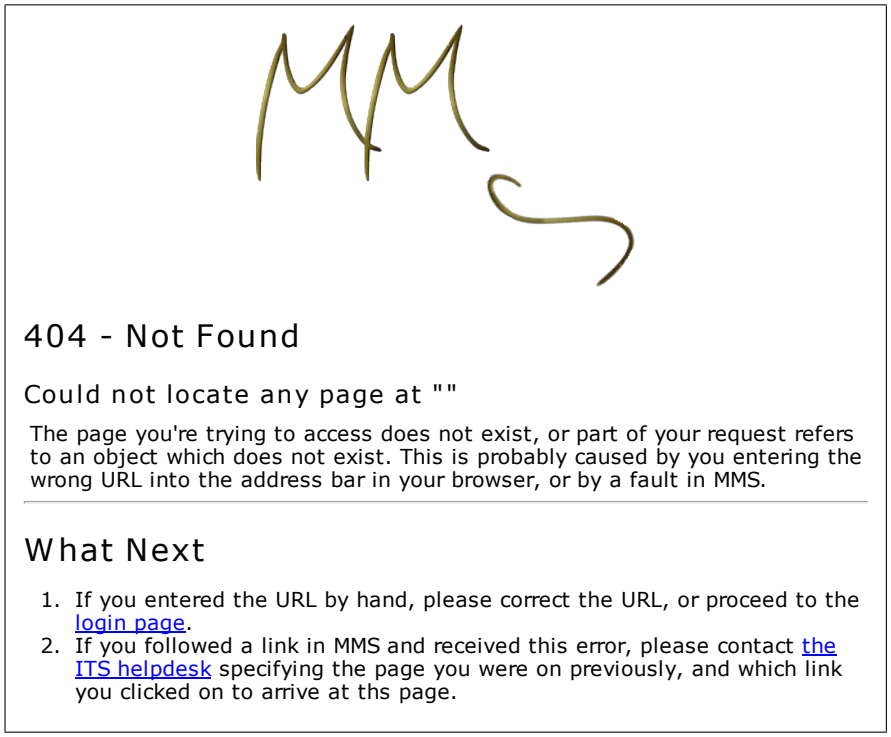


Figure 5.8: Sample Error Page

User Error Reporting

In order to quickly diagnose the cause of an error, it can be helpful to have further detailed information from the user about what they were doing when the error occurred, their software environment (web browser, operating system), and any other relevant observations. Allowing users to report errors in this way can also emphasise to the user that someone has been notified of the problem, and are working on a solution.

MMS initially provided forms for users to report errors, on all error pages, including “404” page not found errors, “403” forbidden errors, etc. This was intended to give a unified approach to error reporting, and emphasise that developers were aware of an issue. After implementation however it became clear that, especially as MMS usage expanded, more “nonsense” errors were being reported through these forms. These included, amongst other problems:

- Requests for access from non-users, as in someone who is not student, staff, external, or otherwise entitled to a user account.

- Incomprehensible/garbled reports, ranging from text that contained English words, but without enough structure to make sense, to foreign languages. These reports were generally tended to be from users who were not logged in, and presumed to be from non-users.
- User passwords, essays, or other material not appropriate for an error report.

These invalid reports were inevitably either a drain on staff time to reply to, or could be a cause of user confusion if ignored. With regret the decision was made to limit error reports to 500-series (server-side error) pages only, reducing the number of irrelevant reports at the cost of impaired user experience. Future work in this area could look at the use of more intelligent filtering or integration with helpdesk software to better handle such reports.

5.9 Security Policy Enforcement

The Department for Business Innovation & Skills' 2013 survey of information security breaches states that over half of surveyed educational bodies had experienced unauthorised access to their systems[110], and provides a specific example:

"A software bug at a large educational body in the Midlands led to hundreds of students' personal data being mistakenly handed out to other students. Several days of complaints and follow-up ensued."

It is therefore clear that the design of a system which deals with personal details, grades, attendance and other sensitive data must prioritise security. Beyond the authorisation and authentication layers, there are a number of design choices which enforce security policy, mitigate risks, and expose functionality for managing security.

The security framework follows the design principle of raising issues early and clearly (this is especially true of the validation framework, which as implemented is sometimes overzealous), and depends on the version control and auditing design principles to support it.

5.9.1 User Interface

The Symantec Threat Report (2013) states that cross-site scripting (XSS) vulnerabilities were the type most frequently encountered during their assessment of websites[126]. This has been a consistent trend in web application vulnerabilities, and accordingly XSS protection is one of the key elements of the security framework.

XSS vulnerabilities are the result of incorrectly validated or escaped user input being included in an HTML page presented to a different user on the same site (hence cross-site). This can result in malicious scripts being run on the victim's browser, which are able to perform any action that a script provided by the website would be able to. This includes retrieving session cookies and passing them to a third party (allowing them to impersonate the user via their session), or to act as a key logger[6]. Such exploits could theoretically be used by students to modify coursework grades or deadlines if they can capture credentials from staff (see also section 5.9.3 for second line defence in these cases).

There are a number of approaches to reducing risk of XSS vulnerabilities which have not been used in MMS, such as marking user input data as "tainted" and tracking it in order to ensure it is not presented without validation[42], static analysis of the application to detect vulnerabilities[129][49] or use of a Content Security Policy[7] to restrict execution of scripts.

The Content Security Policy standard and static analysis methodology are both too new to have been adopted for use in MMS, however would be possible approaches to further improving security of the application. Data tainting requires modifications to core Java classes, and was disregarded as too invasive and risky compared to its benefits.

Template Engine

This left the option of focusing on managing XSS vulnerabilities within the rendering layer. Of the widely used template engines available at the time, Java Server Pages (JSP), Apache Velocity and Freemarker were evaluated. JSP does not support automatic escaping of data inserted into templates, meaning that a single developer error in the thousands of individual places where data is rendered could introduce an XSS vulnerability and therefore is consid-

[6] `https://code.google.com/p/javascript-keylogger/` - accessed 8th July 2103
[7] `http://www.w3.org/TR/CSP/` - accessed 8th July 2013

ered unsuitable for this use case. At the time the decision was made, Velocity also lacked this functionality, and was unsuitable [8].

Freemarker was chosen as it allows tool developers to easily mark sections of a template (typically this would be the entire template) to have inserted data automatically escaped. A fragment of a typical template is shown in figure 5.9, showing the use of the "escape" directive to indicate that all content between the tags should be HTML escaped. The two included files here are standard page header and footer content (such as HTML intro, CSS references, etc.)

```
<#escape x as x?html>

<#include "../mms_head.ftl">

...

<#include "../mms_footer.ftl" />

</#escape>
```

Figure 5.9: Template Fragment showing HTML Escaping

This focus on the template engine is not a universal solution to XSS. While the risk of mistakes is vastly reduced by moving to per-template directives to enable escaping by default, some risk remains. There are further risks introduced by embedding content types recursively, for example Javascript or CSS embedded within an HTML page requires to be escaped according to Javascript/CSS syntax and then into HTML, and this increases the scope for mistakes. To date however there have been no reported cases of successful XSS attacks against MMS.

Transport Layer

The other major design decision in mitigating XSS risks was to use Transport Layer Security (TLS) for all content served from MMS. TLS of course encrypts both the request and response, ensuring it cannot be read if intercepted, however it also provides authentication of the server to the client, through the use of cryptographically signed certificate chains.

[8] Velocity since has added support for automatic HTML escaping as of version 1.5.

Universal use of TLS for all content delivered from MMS ensures that content cannot be tampered with while traversing the network. This essentially eliminates the risk of injection of malicious scripts through a man-in-the-middle attack.

5.9.2 Request Parsing and Validation Framework

MMS contains substantially over 100 different views, many of which in turn contain at least one web form. Parsing and validating user input is therefore a very commonly performed task for controllers. From a security perspective, correctly validating inputs is critical to the integrity and security of the system as a whole. For example negative grades must not be accepted, students must not be able to submit files that are too big to store, staff must only be able to enter marks for students they have appropriate roles in relation to, etc.

An application specific request parsing and validation framework is provided which aims to fulfil the following requirements:

- Minimise effort required to parse input from web forms.
- Add consistent error handling (with an emphasis on messages) for serious (non-recoverable) input errors.
- Ensure primary key validation, for example loading users after validating a submitted user ID, to ensure existence.
- Validate entity references with respect to the context of the request, for example only accepting references to assignments within the coursework tool specified in the request.

In comparison to more generic request validation frameworks such as JSR-303[9], MMS' validation framework is heavily tailored to the typical data types and constraints in use within higher education. Instead of using annotated Java data types to indicate format and constraints on a field, MMS uses modular parsers to which the input parsing and validation is delegated, and return the parsed data or throw an exception as appropriate.

[9] `http://jcp.org/aboutJava/communityprocess/final/jsr303/index.html` - accessed 4th June 2013

These parsers are implemented as Java classes, and any additional details on restrictions are provided as parameters to the class constructor method, where other validation frameworks would typically use annotations. As annotations are optional at compile time, whereas method parameters are not, this approach uses the compiler to enforces definition of input constraints.

Parsers are provided for a number of data types, including most Java data types as well as more complex types such as “String suitable for use as a name”, “Grade”, or “Assignment within specified coursework tool”. The use of highly specific parsers has positive advantages in terms of reducing risks of incorrect or inadequate validation, although it increases developer learning curve. Continuing with the example of a “name string” parser, it incorporates rules that are more difficult to model in other validation frameworks:

1. Trim any leading or following whitespace.
2. Assert the input is not empty, nor does it exceed the defined maximum length for the field.
3. Strip any control characters apart from whitespace characters (for example allowing line feed, carriage return, horizontal tab, but removing the bell character).

Parsers also define the error messages returned to the user, ensuring consistency in reporting and handling of input errors. In making the error handling consistent, the parser only reports errors to the user in one way – the strongest rejection required by any input error. In practise this means that any input validation error results in an error page being presented to the user. This is well suited where input is well outside reasonable inputs (for example an impossible value from a select drop-down list), but unsuitable where user error could cause invalid input values (for example accidentally entering text into a numeric field).

This means that the parser is impractical in cases where the user may legitimately mistype input such that it would not pass validation (for example entering a number or a date). The initial design did incorporate elements intended to allow soft-reporting of errors, by having the calling method pass a list for errors to be stored in, however this was not completed.

Future work here could include parsers returning some indication of relative severity such that the concepts “a number entered into a text field” and “a number selected from a constrained drop-down list” can be reflected in how errors are reported back to the user. Some

consideration has also been given to adoption or merging of the tools with a more widely used parsing framework such as the validation API provided by Spring's Validator API, in order to standardise the approach. It is unclear how feasible it is to do so, however.

5.9.3 Change Auditing

Ensuring that the software application is secure is naturally only part of a larger set of systems and processes that require to be secured as well. A compromise to the operating system, hardware, or the users (for example through social engineering) can all impact the integrity of the application, while these risks are beyond the developer's control. It must be presumed that a security breach is inevitable, irrespective of how well secured the software application is. This is evidenced in shifts in how information security is considered, termed "Assume a state of compromised", in which it is assumed that systems will be compromised[75] (although these approaches are focused more on methods of detecting breaches that have occurred, rather than mitigating damage in case of breach).

Accordingly, damage mitigation in case of a breach must also be considered as part of the security design. There is regrettably little that can be done from a software design point of view, in case of a breach in its host environment, however it can be designed so that it is more difficult to cause permanent and/or serious damage from within the application.

MMS follows a design pattern of avoiding change or deletion of sensitive data once entered, instead marking data obsolete and add new data "on top" of old. This results in an audit trail of changes for such data, containing when each record was created, when it was obsoleted where relevant, as well as details of the user who performed each action. Therefore if/when a security breach happens, there is a log of the following:

- What data has changed.
- When the change was made.
- Who performed the change.
- What the previous value(s) were.

It must be emphasised this is not a purely theoretical risk – one specific example involved an academic leaving themselves logged in to MMS on a publicly accessible computer.

The audit logs provided meant it was possible to verify no changes had been made by their account in the risk window, and would have allowed reversal if they had. Without such detailed logs damage management would have been significantly more difficult, if not infeasible.

5.9.4 Data Security and Integrity

Transport security and content integrity in the web application user interface is handled by the web server (Apache HTTPD). However, for alternative interfaces such as e-mail, there is a need to provide similar guarantees of content integrity.

The actual cryptographic signing process itself is handled by open source libraries. MMS' framework provides wrappers around this functionality to handle details such as key location and identity, in order to simplify usage by non-expert developers.

E-mail Signing

The most visible example of this is in cryptographic signing of e-mails sent to students to confirm receipt of coursework uploaded to the system. These e-mails exist to provide independent proof, as well as acting as a verification for the student that the process has worked as-expected. While server-side copies of the receipts are stored for reference, the e-mails are intended to provide supporting evidence in case of a dispute over when, or even whether, a file was submitted. To date, genuine problems are exceptionally rare, with only one confirmed case where a submitted file was lost (this arose due to a race condition triggered when a single piece of work is submitted twice within a few milliseconds). To fulfil this role the e-mails must be stand-alone, and cannot depend on records stored on the server.

To fulfil this requirement, the e-mails are signed following the S/MIME[10] standard. This is an asymmetric key based solution, meaning that end-users are able to use the public key to verify signatures on e-mail messages, without requiring access to the private key (used for signing). In the event of a claim that MMS has "lost" a file, a student can forward this receipt to support, who are able to use the signature to verify the e-mail is both genuine,

[10]RFC 5751: `http://tools.ietf.org/html/rfc5751`

and has not been tampered with. These e-mails also contain extended details about the file (for example to allow identification of a file that has been uploaded then later deleted), and a message digest of the file itself in order to allow verification that the file referred to in the message matches the file on disk.

E-mail signing is also used to reduce risk of phishing attacks, as end-users can verify that an e-mail has genuinely come from MMS, and has not been forged by a malicious 3rd party. Lastly, it also mitigates less serious (nuisance) issues such as students forging e-mails in order to play pranks on each other, for example trying to persuade a class that their lecture has been rescheduled or relocated.

Document Signing

There is a requirement that specific sensitive data generated by MMS (such as results at the end of a taught module) can be archived for future reference and verification. Where previously paper records have been used, and therefore tampering requires physical access to the records, archival of digital records introduces new risk in relation to tampering with archives.

This archival data is therefore prepared as PDF documents. The PDF standard allows for signing (again, using an asymmetric key cypher) of documents, to verify both origin of the material, and that it has not been tampered with since the document was signed.

5.10 Data Rendering

As well as the preexisting HTML templating tools discussed earlier, an application-specific rendering tool has been developed as part of MMS. A significant proportion of all pages within MMS include one or more tables of data. These require to be presented consistently, both in terms of layout of the table and formatting of data contained within, to ensure a coherent user interface experience. The same content may need to be rendered in a number of different formats, or different views of the data presented depending on use-case.

Flex is a rendering engine which provides a table abstraction layer, which can target a number of different output formats including HTML, Excel and PDF options. These formats

are also further subdivided, for example the HTML target has variants for read-only (for student views of data) and jQuery DataTables[11].

Flex does not directly follow from the design principles, but supports exposing data as widely as possible, allowing data in MMS to be delivered readily in a wide variety of forms and formats as needed.

5.10.1 Data Model

Flex's table data model consists of column headings, body and footer elements, along with metadata describing how the table is sorted, a list of columns to be rendered (the view), and optionally a list of statistics to be generated for each column and added to the footer. The table body and footer are represented by a list of rows (maps), with each map associating a column identifier to cell value. Column headings are stored as a single map of columns to labels, and may be rendered across two rows if any column has sub-columns.

The table stores a list of columns representing the outermost columns of the table. For example consider a table with a "Student" column and a "Grade" column. The "Student" column might be subdivided into "Student ID", "Forename" and "Surname" columns, as shown in figure 5.10. Here the "Student" and "Grade" columns would be directly referenced by the table object, while the "Student ID", "Forename" and "Surname" columns are stored as a list contained within the "Student" column.

Student			Grade
Student ID	Forename	Surname	
1101234567	Dave	Smith	15
1112345678	Diane	Jones	17

Figure 5.10: Example Table with Sub-columns

Beyond the common types for table data, numbers, string, and dates/times, Flex also has types for representing URL links, annotated values (such as a number or string with a status icon attached), and most common form elements such as text input fields, drop-down selection fields, submit buttons, etc.

[11] `http://datatables.net/` - accessed 16th July 2013

5.10.2 Rendering Process

All of the different renderers (HTML, Excel, PDF, etc.) perform loosely the same process for rendering a table:

1. Create the table itself, for example in HTML, this involves writing out the HTML tag.
2. Iterate through the list of columns to be rendered. For each column, look up its label from the map provided in the table, and note whether it has sub-columns. Render these column labels as the first header row.
3. Where sub-columns have been found, iterate through the sub-columns, looking up their labels in the map, and render them as the second header row.
4. For each row of the table body, iterate through the list of columns to be rendered. Where a column has sub-columns, use those instead of the outermost column. For each column to be rendered, look up the matching value in the row and render it.
5. Render any conventional table footer, using the same process as for the table body.
6. Where statistics have been requested (such as mean, standard deviation, record count, etc.), calculate those on each column (where possible) and render the results into the table footer.

5.10.3 Render Configuration

By providing table-specific views, Flex simplifies the process of ensuring tables are rendered correctly and consistently. Consider the rendering of a table where some columns are displayed or hidden depending on user privileges. A template based renderer might have separate conditionals for the optional columns in each of the table head, body and footer rendering parts. A natural approach to this is to test on the relevant privilege before rendering the table cells in question, however such an approach risks later requirement changes introducing mismatches between the rendering conditions.

Flex's rendering is driven by a common configuration option (the list of columns to be rendered), which is only specified once by the controller rather than 2 –3 times in view

logic. In having a re-usable rendering layer, any problems in the implementation are also much more likely to be spotted early, rather than risking most tables working correctly but some having subtle corner cases that can easily be missed during testing.

Read-only or read-write is also configured, although in this case by choice of view by the renderer. Again this is defined once and then re-usable code handles the logic of how to render the model, reducing risk of subtle bugs. This can be useful in cases where a table may be read-write for some users, and read-only for others, for example exam marks for editing by academics and auditing by external examiners.

5.10.4 Calculated Fields

One of the frequent pieces of feedback from staff using MMS was that even where they could download spreadsheets from MMS, because the data was all pre-calculated, it was difficult to see how numbers were calculated and required work to create formulae if they wished to experiment with changes to the values. The statistics footer primarily was added to unify code for calculating summary statistics (sum, mean, mode, row count, standard deviation, etc.) of data sets, but also led into the first stages of allowing Flex to provide formulae where downloading in spreadsheet formats (by providing cells that contained instructions on how to calculate their values, rather than values directly).

5.11 Institution Abstraction Layer

Institutional data and processes are both of significant complexity, in many cases requiring specialised training to understand fully. This can be seen in student record systems such as SITS:Vision, which contains significantly over 2,000 database tables, many of them with over a dozen fields. Further, clear role separation (such as exams, admissions, etc.) in central administration is used to reduce training required to manageable levels. Accordingly, there is a need to simplify interaction with the institution, such that developers can readily make use of data without requiring an in-depth understanding of the institution.

The value of MMS is related to the size of its audience or user base; the more users, the more time it can save, and accordingly the higher its usefulness or value. This audience

is currently restricted to those involved with the University of St Andrews, however one of its design considerations has included the need to support other institutions. This is particularly challenging as MMS is inherently shaped by the University of St Andrews' requirements.

An institutional abstraction layer is provided in order to address these needs. This layer provides a common API for accessing and updating institutional data systems. It has three key pieces of functionality:

- Manages specifics of which data source(s) contain which items of data, aggregating data from multiple sources where needed.
- Simplify interaction models to better reflect intuitive models of the institution.
- Allow for alternative implementations to be provided in a plugin style pattern, for example to change data sources within an existing institution or to allow for a whole new institution.

The abstraction layer follows the design principle of modelling the institution, and further supports acquiring data as close to the original source as possible, and analysing data quality. It is to some extent antagonistic to the principle of exposing implementation detail, as it inherently abstracts away such detail, and a balance must be found through careful consideration.

5.11.1 Implementation

The first version of the abstraction layer consisted of a number of Java classes that provided type-safe primary key types, such as student ID, staff ID, etc., and a single service class that represented the institution and provided methods for retrieving data from it. One of the complexities in retrieving data is that the institution does not have a single data source for many pieces of data, for example while theoretically the student record system has the most accurate record of students on modules, there is a separate database that is used as students' module choices were confirmed by academic members of staff, which was more up to date on occasion.

The most appropriate source of data can also change more permanently, as well as temporarily. During the majority of MMS' development there was no central identity management service for the university, and as such student and staff details had to be retrieved from different data sources. These sources were a combination of the data warehouse, student records, and an LDAP directory. Once an identity management service was brought in, it became the most correct data source.

The abstraction layer allows these details to be hidden, with the institution plugin providing the logic on which data source(s) to refer to specific data items.

In rare cases data sources may conflict with each other, and require manual intervention from staff to resolve. To deal with this the abstraction layer can return not just a simple data set for any given query, but instead multiple data sets with metadata describing the confidence level in each data set. The data sets are then presented to the user for checking and acceptance before applying to MMS.

5.11.2 Data Model

Entities at the core of the abstraction layer include modules, students, staff, units, academic years, semesters and teaching weeks. Some assumptions are made of basic common elements to any institution, as they relate to MMS:

- Entities such as students and staff have primary keys (similar to student and staff IDs as they're used at St Andrews).
- Users may be students, staff, both or neither.
- Students are enrolled on taught modules (or courses, depending on preferred terminology).
- Modules belong to organisational units (as in academic schools).
- Modules are run to a schedule which includes an academic year and semester.

Each of the core entities has a primary key class (in Java). An institution class exists, which contains the core logic for interacting with an institution, including details of how primary

keys are parsed from central records. It also handles more unusual cases, such as parsing of damaged keys or keys taken from swipe cards (where there may be padding around the relevant data). Damaged keys can occur in cases such as where the key is zero-padded, which a spreadsheet might automatically trim.

This use of individual classes for each primary key type ensures that the types cannot be mixed up, as well as providing clarity in method signatures. Consider a method (provided by the institution class) which is intended to retrieve a student's details from central records based on their student ID. Without a specific primary key class, this method's signature might be "getUser(String studentID)", with a specific class it is "getUser(StudentID)". The latter is clearer, as well as significantly more difficult to pass an invalid value to.

The entities themselves each have an appropriate Java class as part of the institution API, although individual institutions can (and are expected to) extend these to add institution-specific data and logic.

Later revisions of the abstraction layer have significantly expanded the data model to include degree awards and courses, groups underneath modules, supervisors for postgraduate research students and disability requirements of individual students. The specifics of these data objects are beyond the scope of this thesis, however the underlying principles used are the same; type-safe identifiers, generic classes for each entity with institution-specific extensions implemented in subclasses as required.

5.11.3 Abstract Marking Scales

The importance of correct and accurate processing of students' marks is a critical requirement of any system intended for use in higher education, given the potential impact of any mistake. This processing can include operations such as calculating averages (weighted or otherwise), translating between different marking scales, or representing marks in a scale-appropriate manner in a chart.

MMS tools that worked with marks (coursework, exam, final grade calculator and quiz) had initially been designed to use double precision floating point numbers to hold marks in memory. Floating point calculations, as implemented in Java and most other programming languages, can introduce errors[35] in the resulting values. These errors are generally

significantly below the precision of the source data, however as a consequence significant development effort is required to ensure that the marks are rounded to the correct level of precision both during calculations and for display.

The use of simple numerical data types meant that any incorrect use of a value (for example comparing marks on different scales) could be difficult to test for (especially for marks in the same range but with different precision). Further, there was a requirement for MMS to support non-numeric marking schemes for some modules, such as staff training modules tended to produce a pass/fail result rather than a finer grained outcome.

Graphing tools designed to present context-appropriate breakdowns of marks (for example by degree class, for the final grade tool) had no insight into the nature of the mark they were presenting, and as such were essentially useless for any cases where the normal 20-point grading scale was not used.

Goals

A new mark API was designed to bring in four key changes to handling of marks:

- Store numerical marks in a fixed-precision manner to eliminate risk of rounding errors.
- Use individual types for marks on different scales, as well as different precisions on the same scale.
- Avoid inappropriate rounding of mark values during calculations.
- Support non-numeric marks (for example pass/fail, red/amber/green, A –F, etc.)

Numerical marks are modelled as fixed-precision by representing the mark using an integer value. This value is the mark divided by their lowest unit of precision. For example a grade 14.5 in the St Andrews 20-point grade scale (a precision of 0.1 steps) would be represented as the integer 145. Using fixed precision numbers in preference to floating point avoids much of the complexity and risk in calculations involving floating point numbers, such as precision related errors.

Introducing type-safety to marks through the use of individual classes for each mark scale reduces risk of accidental mix-up of marks, such as trying to present a percentage mark directly from coursework, to the final grade tool without converting it to a 20-point grade first.

Calculations on numeric marks were changed to use fractional representations of marks during calculation. These consisted of a pair of integers, for the numerator and denominator. When resolved to a single value and rounded, these represented a value within a marking scale, as per the fixed-precision values used elsewhere. As an example of where this was useful, consider a calculation which averages three grades 13, 14 and 13; this should result in a value of $\frac{40}{3}$, rather than $13.3333333...$, such that multiplying the value by 3 results in a value of 40, instead of 39.9. This can be relevant where a set of grades is averaged, then multiplied by a weight factor before being collated with other grades.

5.11.4 Marking Scales

Final marks from taught modules at St Andrews are standardised on a non-linear 20 point grade scale, however within each course coursework and exams are often assessed against other scales and then converted to a 20 point grade. This is particularly true in science subjects, with percentage marks being awarded for assessed work, averaged to provide an overall mark and then converted at the end of the module before being passed into the grade aggregation tool.

A small number of taught modules, which are not part of graduating degree courses, do not return 20 point grades, instead resulting in a simple true/false boolean outcome. Lastly, while research students typically do not have coursework in a conventional sense, they may have deliverables which are assessed on a "traffic light" system of red/amber/green (representing confidence in the student's progression to date).

API

The API is based on the factory design pattern, with tools interacting with a number of Java interfaces. The "Mark" interface provides methods for common operations on marks including calculating mean mark of a collection of marks, rendering the mark to a string

value for display, and indicating common ranges of marks for use in graphing (for example by degree class for 20-point grades). Additional functionality for numeric marks is provided by a common superclass, NumberBasedMark; this includes methods for determining precision, minimum, and maximum values.

Modules, tools, schools and other entities which deal with marks store their mark format by referring to the full qualified name of the relevant factory implementation. The MMS framework handles instantiation of the factory class using reflection. Implementations of the "Mark" class support marking scales including:

- grades A-F
- grades on the University of St Andrews' 20 point scale
- percentage marks
- 20, 40 and 100 point marks
- pass/fail boolean marks
- red/amber/green ("traffic light") statuses

It should be emphasised that a 20 point grade as referred to at St Andrews was distinct from a 20 point mark, as grades are not on a linear scale (a 10 is not half as good as a 20, for example), while marks are. As such while they are both represented as numeric values between 0 and 20, they cannot be used interchangeably. One of the safety features introduced by this API was this strict separation of types, using Java's strongly typing to enforce isolation of different implementations. Accidentally mixing mark types results in a run time error, as each class casts "Mark" implementations into its own class when parameters are passed in.

Subclasses of number based marks such as percentage or 20 point grades were further subclassed to provide separate representations for persistence, and for use in calculations. The former used a single integer value to represent the mark, and could not store any further precision, the latter stored a mark as a fraction using two integer values such that any division and multiplication of the marks was done without any premature rounding.

Mark factories enumerate all possible values of their marks into arrays, and when translating a string or numeric representation (for example from a database) to a mark, access

values inside this array. This provides a range check on marks enforced by the Java virtual machine itself. For example of a value of 20.5 was loaded as a 20 point grade, it would be represented as an array index of 205 within an array limited to 200 entries, causing an exception to be thrown. This acts as a further safety net in case range checks were missed elsewhere.

Parsing

This mark abstraction layer could also be used as part of the request parsing framework (see 5.9.2), enabling it to automatically configure itself to parse marks as appropriate based on the configuration of the module, tool, etc. the view belongs to. In the current implementation the lack of support for returning errors to the user inline with the original page means this is infeasible (returning separate error pages to the user for reasonable input errors would be extremely user-unfriendly), however.

5.11.5 Mark Conversion

It is common to require to translate marks from one scale to another, for scenarios such as converting percentage marks awarded for coursework to grades for reporting at the end of a module. A separate utility API is provided which performs these mark translations. This API is configured using conversion tables which indicate where marks on the source scale relate to the destination scale. Scale points between specified mappings can be resolved either as step-wise (so the target mark is the highest mark specified in the map, with a source value below the actual mark), or interpolated (so the distance between specified source marks is translated to the distance between target marks).

Source (%)	Destination
20	5
60	15
800	18
100	20

Table 5.1: Example Mark Translation

As an example consider the translation map shown in table 5.1. With step mapping, a mark

of 20% would give a grade 5, but a mark of 40% would also give a grade 5. On the other hand, with linear mapping, a mark of 20% would still give a grade 5, but a mark of 40% would give a grade 10, as it is halfway between the two grade points. This mapping is designed to provide enough flexibility to allow for the grades students are awarded to be modified if an exam is felt to have been unusually easy or hard.

This translation API currently only works for numeric marks, as translation between non-numerical mark types was not a requirement and so no use case was available to test against.

5.12 Server-Side Scripting

In order to enable rapid development in light of frequently changing requirements, MMS has tools which support server-side scripting in the form of Javascript. This scripting functionality allows staff to write code which is executed within MMS, eliminating the need for downtime to deploy new code, while reducing risk of errors as result of communication over requirements. Details of where and how scripting is used is further detailed in the User Tools chapter.

The scripting functionality provides unit testing support to encourage code quality, as well as to give staff the confidence that implementations are correct.

Server-side scripting support follows the design principles of exposing implementation detail (and is in fact by far the best exemplar of this principle), with the unit test support designed to ensure any issues are raised as early as possible.

5.13 Tool Data Source Interfaces

One of the strengths of MMS is its ability to readily reuse its own high quality data as the inputs for other tools. It is however desirable to isolate tool developers from the implementation specifics of other tools, in order that the data consuming tools can use any suitable tool without knowing implementation details, and that data providing tools are not overly constrained by a requirement to maintain implementation compatibility with previ-

ous versions. To achieve this, interfaces are defined by the MMS framework for a number of different scenarios.

The main example of this approach is the interface for passing marks between tools, with interfaces also provided for attendance (lecture or tutorial) and student engagement.

The abstract interfaces enable tools which follow the design principle of exposing data as widely as possible, making it easy to re-use data within tools without requiring an understanding of the tool's implementation.

5.13.1 Marks and Grades

The grade aggregator tool consumes marks and grades from other tools in order to calculate final grades for students. Its original development was done at a point in time when the only suitable data sources were the coursework and exam tools, however the use of a defined common interface (as opposed to developing against the tool implementations) has allowed data sources to expand to support a number of other tools.

These later tools primarily were the quiz tool and a WebCT grade importer. A number of incomplete prototypes have also been developed but never completed, including a Moodle grade importer and a spreadsheet-based data source for holding data imported from arbitrary sources.

The grade aggregator itself then implements a further interface for exposing its calculated grades. This interface is specifically designed to feed grades from modules, rather than components of a module, and is consumed by the institution plugin. As with the intra-module grade source interface, this was developed as part of future planning, although unlike that interface it has to date seen only one implementation.

In comparison to similar mark APIs in Learn or Moodle, MMS' interface is richer, providing better support for scripting of the data processing. For example the final grade calculation scripts can examine component marks as well as the overall mark for coursework, in order to make decisions. This enables modelling of complex behaviour such as bonus marks awarded for consistency of quality of work.

5.13.2 Attendance

Attendance data is relevant to use cases such as monitoring students for early indications of problems with their academic engagement. With two tools providing this data (the lecture and tutorial tools), a common interface means consuming tools do not require to understand the tool details. This also adds scope for integration with external systems, such as academic timetabling systems which could use this information to revise room requirements based on actual attendance.

5.13.3 Student Contact

The last interface slices across a number of different use cases. As part of changes to the requirements for institution Highly Trusted Sponsor status for tier 4 visas (adult students) in 2011, UKBA requires higher education institutions to record and report on student contact with the institution. Contact in this scenario is well defined, and covers a number of events in the normal student's academic lifecycle including matriculation at the start of term, submission of coursework, and attendance at lectures and/or tutorials.

Matriculation is not currently managed by MMS, however coursework submission and attendance are, and accordingly the relevant tools can provide this data for reporting to central records, and onwards to the UKBA.

While the attendance tool could in theory provide this data directly, the provision of a generic interface allows for future extensions to the reported event types. For example if quiz completion was counted then the quiz tool could implement this interface and would automatically be included in reported data. See also discussion of the journal tool in the Future Work chapter, under section 12.5.6.

5.13.4 Coursework Submission

As part of the LAVA project (discussed in detail in chapter 10) tools were developed which produced assessed work as a result of student input (the collaborative form). A generic interface for tools which could accept coursework submission (the coursework tool is the only current implementation of this interface) was developed to allow the collaborative

form to submit to an assignment, ready for marking by staff.

5.14 Conclusion

MMS' architecture is designed to provide a robust yet simple to use support framework for its hosted tools. The use of a layered approach to security, based on the organisational structure underpinning the authorisation model, moves much of the responsibility for security away from the the tools and into shared framework elements, reducing the scope for errors in design or implementation. Further, the hosted tools are not required to understand the complexities of the authorisation model, further simplifying the development process.

The architecture uses a fail-fast approach to error handling, and ensures that any errors are reported quickly and clearly to support staff. This, in combination with a version controlled data model, ensures that any issues are resolved quickly with a minimum of long-term impact on the overall system.

These factors combine to enable and encourage rapid development in response to changing requirements, while effectively managing the risks inherent in such an approach. The isolation of developers from low-level concerns allows them to focus their effort on end-user functionality, and enables development of tools which may be impractical otherwise.

Chapter 6

Integrations

In reducing administrative overheads one of the most obvious routes is to eliminate duplicated effort. Any occurrence of manual entry of data which is already known to the institution is an opportunity to eliminate such effort. To this end, we consider the following:

- What data does MMS use for which there are existing data sources, and what further data could MMS used from these data sources?
- What form do these existing data sources take and how can we adapt them where needed?
- What data within MMS can be re-used by other external systems?

Further, we will consider how data within MMS can be used to drive processes in external systems.

MMS collates a substantial quantity of data from a variety of sources. Exposing this data for wider use increases the data's utility, and provides additional motivation for time spent improving data quality within MMS. In contrast to a conventional data warehouse, MMS acts as an intelligent interface to the data, providing an authorisation layer, difference generation, and other supporting tools to simplify integration with external systems.

These integrations embody the design principles of acquiring data as close to the original source as possible, of exposing data as widely as possible, and of course of integrating with

external systems. They are enabled by the design principle of modelling the institution, which ensures the model within MMS is generally compatible with other systems.

6.1 Data Sources

Configuration of MMS requires significant quantities of data which have existing sources including student and staff identities, details of modules running, lists of students enrolled on modules and student grades. Student grades include both assessment outwith MMS, and final module grades after any adjustments made after grades are reported from MMS (i.e. by the relevant Dean).

It is useful for MMS to also collate and present data from external systems, in a similar model to a web portal such as uPortal, and specifically reading lists. Lastly, as learning and teaching tools have been added to MMS it is desirable to be able to import pre-packaged content from external sources.

6.1.1 SITS:Vision

Early versions of MMS used the central data warehouse as the standard source for student records such as student details, course choices, degree intentions, etc. However the data warehouse is an interpretation of data from SITS:Vision, refreshed daily. As a result the data suffers from lack of detail, potential issues in interpretation, and during periods of high rate of change can be significantly out of date.

Direct integration of a data feed from SITS was added to resolve these issues. Data is read in via SQL, wrapped within the institutional abstraction layer. Recently SITS has introduced web service support (StuTalk) for integration with other, and the code is currently being updated to use these officially supported interfaces. This data includes student identities, modules being run, and enrolments on modules.

SITS maintains a highly detailed model of the institution, and mapping this to a conventional web application may require re-interpretation (as was done by the data warehouse). In the case of MMS however its use of a model which closely reflects the institution limited this requirement, although some translation was still required, such as filtering records to

eliminate those not of relevance to MMS (i.e. students who have not yet started).

6.1.2 WebCT

St Andrews initially chose WebCT as its official VLE in 2001 (later replaced by Moodle in 2010). Although MMS had some limited VLE-like functionality, in contrast WebCT was intended as a VLE and included more powerful content and discussion tools, as well as a quiz tool that was popular in a number of schools (including Geography, International Relations, Modern Languages and Physics), both for assessed work and self-assessment.

Where WebCT assessments formed part of a student's final grade for a module, there was a need to aggregate the assessment results with other assessments (coursework, exam, lab works, etc.). The final grade tool (see 7.5) was rapidly becoming the preferred method for performing this data aggregation, however this meant the data needed to be copied from WebCT to MMS.

The introduction of a WebCT grade import tool automated this process, reducing risk of mistakes while ensuring the shown data in MMS reflected the latest data from WebCT at all times. This tool was exposed to the final grade tool via the genericised interface (see section 5.13.1), enabling it to treat it as any other data source.

6.1.3 Moodle as a Data Source

With the replacement of WebCT by Moodle, there is now demand for import of grades from Moodle. To date staff have been encouraged to use MMS to store grades and assessed material wherever possible, due to its support for auditing of changes, and as such this functionality has not been developed.

If it was considered desirable, however an integration could be provided using the functionality to "publish" grades from Moodle, which involves issuing a static URL where the grades are located and can be repeatedly retrieved from. This URL could be used in place of directly uploading the exported file, as a way of allowing MMS to automatically refresh grades.

6.1.4 Talis Aspire (Reading Lists)

Details of reading lists for modules are stored in Talis Aspire[1], an application managed by the university library. Exposing these reading lists via MMS enables students to avoid switching between systems, while reusing enrolment data to ensure the correct audience.

6.1.5 IMS QTI Support

As demand on VLE-like function support for MMS increased, there was a need to start considering prepackaged content. The migration project from WebCT included a substantial number of quizzes that required migration, and as WebCT provided the ability to export those quizzes to the IMS QTI format, it made sense to add support to MMS. Limitations in the implementation of the quiz tool mean many common question types cannot be supported at this time, including short written and essay question types, however this could be addressed in future work if there was sufficient demand.

6.2 Data Consumers

MMS contains a wide variety of high quality data of potential relevance to a number of external applications. Exposing this data via database links or web services reduces duplicated effort, increases reward for maintaining MMS records, and can improve functionality of external systems.

6.2.1 SITS:Vision

Once final grades are calculated within MMS, they need to be returned to central records. The initial interfaces for performing this task have used CSV (comma separated value) files or using SQL to insert records directly into the SITS:Vision database. The former required significant manual handling, the latter bypassed triggers and validity checks within the SITS:Vision application.

[1] `http://talisaspire.com/` - accessed 23rd July 2013

The recent addition of StuTalk web services to SITS:Vision means MMS can now write to the system is a better supported form, and work to integrate with this interface is ongoing.

6.2.2 Mobile App Support

Mobile applications for higher education to date tend to focus either on delivery of fairly general information, or on mobile learning (frequently referred to as m-learning). oMbiel's campusM product and Straxis' u360 mobile products are examples of the former, both providing information such as staff directories, campus maps, etc. For examples of m-learning, see Herrington and Herrington[46] and O'Malley[19].

These are unusual areas to focus on, given that mobile devices tend to be relatively limited in terms of display size, input options, processing capability, etc. It would seem that there is more of a need to focus on functionality which mobile devices are uniquely placed to support, namely delivery of time sensitive information, such as coursework marks and announcements

MMS contains a significant amount of such data. It also has tools for which there is demand from students to access them as early as possible, such as the tutorial signup tool. There is significant demand for places on tutorials at convenient times, and as places are assigned on a first-come first-served basis, students already frequently access the tool from browsers on mobile devices. If m-learning was considered a desirable goal, it would also be possible to provide access to course content.

Such data and tools suggest opportunities for mobile applications to provide improved interfaces (compared with a web-only interface on a mobile device). A partial mobile application prototype has been developed for some of this functionality, however unfortunately it has not been completed due to time constraints. However, others have managed to make further progress towards this goal.

MSc Student Projects

Some prototype work was done on the St Andrews mobile application to adapt it to retrieve data from MMS via web services, however development time constraints meant it was not

taken beyond the prototype stage. These web services remained as part of MMS, and were later used for two MSc projects.

Two MSc students independently developed mobile applications which integrate with MMS as part of their dissertations. Sulaiman worked on an application for the iOS platform [125], with Luktuke working on Android[73]. Both students consumed data contained in MMS in some manner, although Sulaiman's work was more generalised in its intended use cases.

Their designs are naturally their own work, and can be considered to reflect a student's own view of what functionality mobile application integration for MMS should provide. The students were supervised by Colin Allison, with assistance from myself.

Both provided fairly similar functionality in terms of integration with MMS, providing a list of taught modules a student was taking, and allowing them to retrieve their own current grades. For the Android version the original design called for providing push notifications (based on a publish/subscribe model[24]), although regrettably technical issues compounded by the replacement of the C2DM middleware[2] by GCM[3] meant could not be completed in the time available.

These designs can be considered a good starting point to any later analysis of use cases for mobile applications integrating with MMS. See also 12.5.4 in the future work chapter.

In terms of integration with MMS, where the applications required to retrieve data (as opposed to having it pushed to the applications from MMS) this was relatively straightforward. In particular as the web application interfaces implement authentication and authorisation based on the end-user's credentials, rather than the API not having any inherent restrictions (as is the case for WebCT's web service interface), there was no risk to system security from allowing students to work with these interfaces.

6.2.3 Google Search Appliance

Given students are expected to use a number of different systems (MMS, Moodle, potentially school-specific systems) for module-related tools and material, inevitably there will be some confusion over which system to use in different contexts. MMS exposes its search

[2] `https://developers.google.com/android/c2dm/` - accessed 3rd June 2013

[3] `http://developer.android.com/google/gcm/index.html` - accessed 3rd June 2013

interface via the OneBox[4] interface, enabling a Google Search Appliance to pass queries to MMS for it to address.

Currently this is limited to matching modules by module code for simplicity. A matching integration has also been added to Moodle, meaning that one search identifies matches in both of these systems, as well as any other matches within publicly accessible institutional web pages. With additional resources it would be possible to expose course content such as lecture notes to the search appliance, enabling students to search the content by keywords and similar.

6.2.4 EvaSys (Module Evaluation)

MMS now contains the most consistently accurate data staff involved with teaching of modules, as well as students enrolments. As a result, its data is ideal for use in producing surveys for evaluation of teaching of modules and ensuring those surveys reach the correct audience. These surveys are generated by the EvaSys[5] software. This data is exposed as an archived set of XML documents which can be retrieved on demand from a web service.

6.2.5 Moodle

Moodle requires data on staff and students associated with each module, and with this data occurring in MMS as well as central records obviously it makes sense to reuse existing data sources. Taking student data from SITS or the data warehouse would mean Moodle reflected the students taking a module – a logically obvious solution, but failing cases where students were allowed to "audit" on modules they weren't officially taking, or similar unusual cases not reflected in official records. MMS already stores detail on student enrolments on modules sufficient to differentiate these auditing students from others, making it logical to re-use that data.

MMS also pushes details of running modules into Moodle as part of the module activation process, this is discussed in detail in the chapter 7.

[4] `https://developers.google.com/search-appliance/documentation/614/oneboxguide` - accessed 29th July 2013

[5] `http://www.evasys.co.uk/start.html` - accessed 30th July 2013

6.3 External System Management

Beyond the straight-forward source/consumer relationships detailed above, there are several external applications which can make use of data in MMS but do not necessarily reflect that data, provide functionality to MMS, or otherwise integrate in more complex forms.

6.3.1 Turnitin

Turnitin has been adopted as the preferred plagiarism detection tool at St Andrews. Accordingly, there is a significant quantity of coursework which must be submitted to Turnitin for evaluation. There were three basic options for the submission of work to Turnitin:

1. Submission directly to Turnitin by students.
2. Submission by staff on behalf of students.
3. Submission via an integrated coursework tool.

The first two options fail to reuse data which is already present, requiring staff to set up and manage records in Turnitin in addition to MMS. Naturally integration with the coursework tool in MMS is therefore the obvious option, and eliminates what would otherwise be a significant administrative tasks.

The integration uses what Turnitin developers term “deep integration”, in that Turnitin is wrapped entirely by the user-facing application, with only the detailed report on a piece of work presented by Turnitin. The advantage to this approach was a seamless integration and further that MMS handled all submission to Turnitin.

Submission to Turnitin is done by a background task which runs regularly (around once every fifteen minutes), sweeps for coursework in a suitable file format which has not yet been submitted, submits it to Turnitin, and records the resulting Turnitin object ID. A second background task then uses these object IDs to fetch the resulting report from Turnitin. This decoupled submission ensures that coursework can be submitted to MMS even if Turnitin is unavailable, as happened for in 2012 due to extremely high load[101].

Suitability of files for submission to Turnitin is determined by detecting MIME type of the file using a combination of the MIME type provided by the user's browser during submission and "magic bytes" detection based on the file contents. Staff do have the option to restrict acceptable file name extensions for a piece of work, to reduce risk of incompatible formats being submitted by accident.

More complex file handling was considered, for example automated conversion from incompatible formats to compatible formats, or extraction of documents from within Zip archives, however it was considered that this was likely to introduce undue risk of error (for example problems introduced during file format conversion).

A breakdown of Turnitin usage is included in evaluation (see section 11.4.4) and illustrates the workload which is avoided by pre-emptively automating this process.

6.3.2 Virtual World Administration

From the academic year 2007/8 St Andrews started looking at potential for use of virtual worlds in education. Second Life was favoured for a number of reasons, including integration options with external web services and its easy to learn scripting language (LSL). Virtual worlds bring with them a lot of the same administrative issues that apply to the real world. Attendance needs to be tracked at lectures and other events, coursework needs to be submitted to tutors, coursework materials need to be delivered to students, etc.

Additionally, tasks that are normally infeasible to automate in the real world are feasible in virtual worlds, for example setting up experiments for labs, tracking student progress through exploration-orientated coursework and recording content of meetings. There are also tasks unique to administration and virtual worlds (this list reflects the most significant issues encountered, and is not intended to be exhaustive), including:

- Tracking of virtual world identities in relation to real identities, for the purposes of attendance monitoring, course involvement, coursework submission, etc.
- Ensuring students are in the correct place in-world, especially when moving around in-world, for example doing "digital field trips" where groups of students may be taken around a number of areas not managed by the institution.

- Dissemination of teaching materials in-world, for example sample objects, scripts, etc.
- Management of teaching/experimental spaces, for example resetting in-world experiments between lab sessions.
- Tracking coursework submission in-world, for example scripting exercises.

The Sloodle[6] project (sponsored by EduServ) provides an integration between Moodle and Second Life, however it is focused on teaching aspects rather than administrative.

Account Creation

The first task for introducing students and staff to a virtual world is to create an identity, or account, for them. While virtual worlds provide their own tools for this, web service interfaces such as the registration API (RegAPI) for Second Life allow institutions to provide custom interfaces for account creation.

Integrating MMS with RegAPI means details such as e-mail address and date of birth can be automatically provided for users, as well as ensuring users start ”Minerva Island“, the university's private island in Second Life. The LLSD[7] (Linden Labs Structured Data) library developed for this integration is currently the reference implementation for Java [8].

In creating the accounts, MMS can also record the UUID (universally unique identity) of users, allowing teaching staff to identify students in-world. Currently there is no option to associate pre-existing accounts with MMS users, however there is no technical reason this could not be added.

As of the time of writing, RegAPI has no provision for administrating accounts once they are created, nor for performing authentication using an external authority. This can be problematic if users have a problem with their account, as they logically expect the system that created the account to be able to make changes such as password reset.

[6] `http://www.sloodle.org/` - accessed 23rd June 2013
[7] `https://wiki.secondlife.com/wiki/LLSD` - accessed 14th July 2013
[8] `https://wiki.secondlife.com/wiki/Linden_Lab_Official:Reg_API_Examples`

Coursework Submission

One problem highlighted by early experiments with coursework in Second Life was how to submit in-world assets for assessment. Content protection or digital right management (DRM) in virtual worlds such as Second Life can restrict the permissions users have with content they receive from other users. By default content created in Second Life cannot be copied or modified by anyone apart from the original creator, and students often did not change these settings.

As a result, student coursework that attempted to create copies of itself as part of its normal operation failed when passed to the tutor for marking, as it was considered "unlicensed duplication" by the DRM. Lack of modify permissions also meant that teaching staff had no ability to read source code (which is locked under the security model), or to attempt to repair the damaged submissions[7][103].

While instructions were provided on protocol for submission of virtual world assets as coursework, students were used to a more formal and guided process, and a one-off alternative submission process appeared to cause significant confusion. From an administrative point of view, submissions lacked clear timestamps for when they were submitted, and students had no submission receipt. There is a clear need for tools to support coursework submission within virtual worlds, for example verifying permissions on work before submission, as well as providing receipts for work as MMS does for submission of work as files.

Outcomes

Use of virtual worlds with groups of undergraduate students at St Andrews remains generally low, typically comprising part of one or two modules in each academic year. There is further activity with groups of prospective students as part of open days, and in research (especially postgraduate research). The preferred virtual world platform has since been replaced with the open source OpenSim, which provides significantly more flexibility as the underlying engine can be modified by the researchers[3].

6.3.3 Facebook

Smith & Caruso report that over 90% of all students use Facebook, and over half use Facebook to communicate with classmates[122]. There have been a number of studies looking at Facebook as a learning tool[105] or integrated into learning tools[130], however these have focused on the learning and teaching aspects rather than administration. The College Knowledge Challenge, a $2.5 million fund for development of "innovative apps on Facebook that will help students apply to, attend, and stay in college."[9] funded a number of applications related to administrative tasks such as application to universities, comparing costs of different universities and transferring between universities. While these applications do assist with administration, they relate to tasks MMS does not currently support.

Partial Facebook support was integrated as an experiment to consider how social media platforms could be used to improve communication and engagement. It had been hoped to provide a socially-relative feed of information, for example prompting students to announce details of modules and tutorial groups they are attending, allowing their friends to schedule based on that knowledge. Coursework-related status updates could also be created automatically, ready for the student to post with a single click, for events such as "Got my CS1002 coursework in on time!" or "I got a 2:1 grade in CS1004". It is hoped that including the students' social groups in their academic activity could potentially improve engagement with the material.

It also became clear from discussions with students that they would have concerns about allowing an institutional system to access their Facebook account. Given examples where institutions have taken actions against students based on material posted to Facebook[59], there is clear evidence to support caution specifically with regards to the institution having such access. As with any integration which entrusts personal information to a third party, it is also important to be confident that the data will be securely stored, and careful consideration of Facebook's security provisions would be required.

A prototype integration was developed to allow users to associate Facebook accounts with MMS accounts, however it became apparent that the integration required both substantial modification to MMS, and changes to the security model. Given reluctance from the students to mix institutional systems and Facebook, and the task providing significantly more

[9]Taken from `http://www.collegeknowledgechallenge.org/about/` – accessed 6th July 2013

complex than anticipated, the decision was taken to defer further work on these features. There is scope for future work in this area, and in particular looking at use of data in a social context (both to improve the student experience or for motivation).

Chapter 7

User Tools

This chapter will describe the end-user functionality presented by MMS, including its design rationale and software architecture. Experiences gained during the implementation and refinement of this functionality will also be discussed. MMS presents a number of different tools intended for use by students and staff, which this chapter will describe. Tools in this context generally are associated with a module or academic school.

In evaluating what functionality was required, and how best to fulfil those requirements, a number of questions are considered:

- What administrative activities consume most of an academic's time? Further, what activities consume most of administrative staff time?
- What is it about these activities which leads to them being time consuming?
- Are there any clear opportunities for streamlining these activities?
- How can we reduce risks in these activities?
- What tools and processes are already in use for these activities?

In seeking to reduce administrative overheads, looking at where these overheads are most significant and any apparent approaches to reducing them is an obvious first step. Looking specifically at ways to reduce risks can further reduce overheads by removing steps which require significant care and attention, as well as work involved in fixing any errors which

arise. For example, in using a single shared source of mark data, MMS virtually eliminates risks of versioning issues in mark data. This reduces time spent tracking versions, manually merging or replacing data, and correcting any problems with these processes.

When looking for opportunities to streamline processes, it can also be useful to look at other's contribution to the process. For example in having students submit work electronically, there is an opportunity to automatically generate accurate coversheets, track versioning and lateness automatically, and overall bring an increased level of consistency to the outcome. It is also important to consider the environment into which new tools are being introduced. Often there are preexisting tools which partially improve processes, and with which staff have familiarity. By adopting similar designs and workflows, it is possible to reduce the impact of changes, while leveraging other's experiences.

With later development, user demand also was a major driver in deciding where to prioritise development effort. Such demand approximates areas which require attention, and in meeting demand it encourages acceptance of the system.

Architecturally tools within MMS are independent components, which integrate with each other only through application interfaces defined in the core architecture. Tools are isolated from each other to simplify maintenance, ensuring that tools can be developed independently with minimal coordination required between developers. The configuration interface has been included in this chapter, although it is part of MMS' framework rather than an independent component.

7.1 Benefits

Using the institution benefits framework referenced in White and Davis[134], the tools in MMS seek to provide the benefits shown in table 7.1. Here the emphasis on institutional and faculty benefits can be seen over the more common learning and student-related focus.

Impact Areas	Specific/ Generic	This Case	Objective
Student Learning	G	e	Enable students to retrieve course content at their own pace
	G	✓	Address needs of students irrespective of location (on/off site or distance learning)
	G	p	Increasing student performance
Student Experience	S	e	Increasing student satisfaction
	G	e	Increasing retention
Institutional Benefit	S	✓	Reduce administration time
	S	✓	Significant reduction in risk of errors
Institutional Indicators	G	e	Improve student satisfaction
Faculty Benefit	S	e	Reduce teaching time
	S	e	Reduce assessment marking time

Key: G: Generic; S: Specific;

✓: explicitly addressed and to be evaluated;

e: consequential benefit which needs to be evaluated

p potential benefit, not explicitly evaluated at this time

Table 7.1: Institutional Benefits for User Tools

7.2 Configuration

Naturally MMS provides tools for managing details of modules, students, tutorial groups and other entities within the system. In accordance with the design principle of acquiring data as close to the source as possible, the primary focus is on automatic migration of data from central data sources, however manual controls are available for virtually all records to handle cases where central data is not available, inaccurate or unsuitable.

From experiences with the INSIDE project it was clear that central data sources had a significant lag in being updated, and as such MMS used its own independent records to allow differences to exist until corrections could be applied to the central source data. This use of local copies of core data also simplifies data loading and allows for database level integrity checks.

The model used meant that there was no "golden source" of data; school-local records could be updated more rapidly, and were often more current, however central records were considered authoritative at the end of the academic year. Central records could not be blindly written over local records, as this could often replace the records with less accurate versions, nor could local data be written over an authoritative source.

7.2.1 Module Activation

Modules are the main entity type around which MMS functions, and as such activation of modules form the backbone of its configuration. Almost 1,700 individual modules are listed in MMS for the academic year 2012/3, and obviously individually setting up this many modules manually would be both significantly time consuming, and error-prone. Accordingly school-level staff confirm modules to be enabled within MMS, and MMS then automatically configures the module. Module lists are extracted from SITS:Vision's records of modules expected to run in the current or future academic years.The interface for module activation is shown in 7.1.

There have been cases where staff have confused editing data in MMS for affecting central records, and therefore it has been important to differentiate between this activation process, and creation of new modules in central records. Hence the term "activated" is used, in contrast to "created" as might be used in other similar applications.

In exceptional circumstances there may be disagreement between the academic school and central records on modules running in an academic year. Examples have included mix-ups in paperwork adding or removing modules from the curriculum, or last minute changes to modules running due to illness.

System administrators have the option of activating arbitrary modules without reference to central records. This is required in cases where students need to be enrolled onto a module or course content set up before there has been an opportunity to update central records. This is also used to provide module-like entities which are not backed by actual modules, for example for deliverables for a cohort of postgraduate research students (see also section 12.3.2 for a discussion on how this could be better modelled in future).

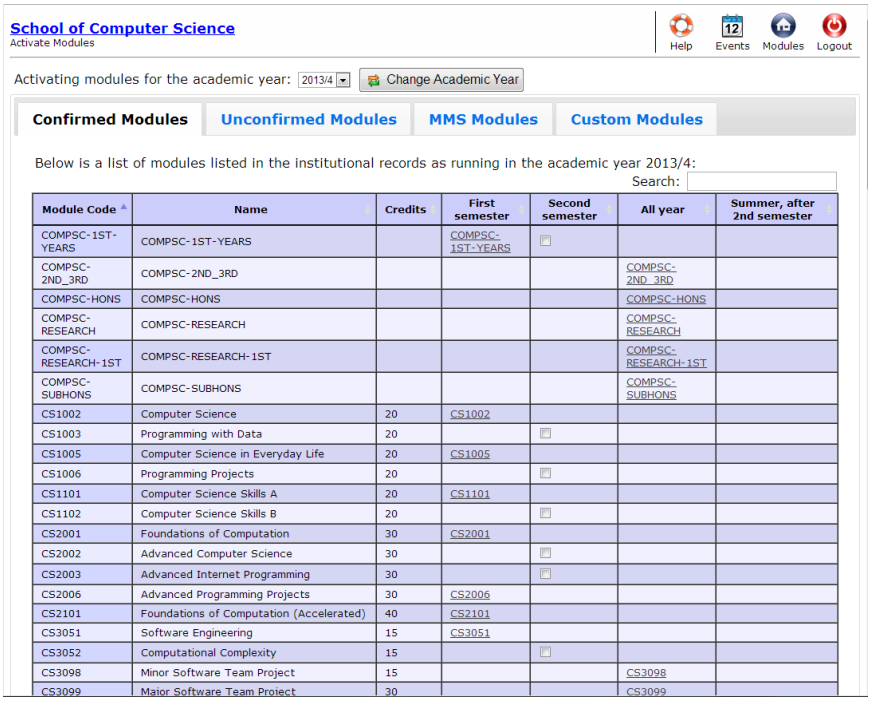


Figure 7.1: Screenshot of Module Activation

Module activation imports students from central records (these relationships then automatically update each night). An optional automated "roll-over" process was introduced due to demand from academic and administrative staff. This process identifies the most recent occurrence of the same module and performs the following additional configuration actions on the newly activated module:

1. Duplicates groups from the source module, to the new copy.

2. Staff enrolled on the source module are assigned the same roles on the new module. Staff on groups within the module are not assigned at this time, as group-level enrolments are likely to change between academic years.

3. Duplicates tools from the source module, to the new copy. Note that configuration is not done at this point.

4. The rollover process is triggered on each tool, where supported, which will then copy its configuration into the equivalent new tool.

Migration of dates between tools is of particular note. Simply choosing the same date (i.e. 6th February) would lead to a change in the day of the week that the deadline falls on. Further, teaching does not necessarily start or end in the same week each year, so an item of coursework due on the Friday of the 1st week of term one year may be on the Thursday of the 2nd week, or even of the week before term starts, if naively migrated. Instead dates are mapped to the week of term, and day of the week, before migration, such that a deadline on the Friday of the 1st week of term is always on the Friday of the 1st week, irrespective of what date that is within the calendar year.

Where applicable MMS will also update records in Moodle and/or Turnitin in preparation for the new module. For Turnitin this is triggered by the roll-over of the coursework tool, and constructs the new module and assignments within Turnitin. The Moodle rollover process is substantially more advanced – a bespoke script has been added to Moodle, to which MMS sends details of the module being activated, and the module from which rollover should be performed. The Moodle-side script then completes the following steps:

1. Creates the new module in Moodle.

2. Identifies the source module, where it exists. Where no match can be found, the rollover is aborted.

3. Runs the Moodle backup process on the source module.

4. Restores the backup into the new module.

5. Re-writes URL links in the new module, where they contain references to specific academic years. Here MMS' use of meaningful identifiers in its address space is

beneficial as it simplifies the process of ensuring addresses can be updated appropriately.

7.2.2 Student Enrolments

Student-module enrolments illustrate many of the design principles of MMS, in that they model roles based closely on institutional structure, acquire data from original (central) sources, provide audit trails and version control for the data, and provide tools for analysing data quality (and highlight any issues). Records are automatically updated overnight from central records (SITS:Vision). A user interface is provided to analyse changes and differences to identify any issues arising, and users can override imported data where needed.

This interface consists of two parts; e-mails sent to staff to notify them of changes, and a web interface for reviewing outstanding differences between MMS and central data sources (such as the Data Warehouse, SITS:Vision, and the Advising database). The web interface is shown in figure 7.2.

Note that student and staff on modules are handled independently, through separate user interfaces, to correspond with the isolation of data records. This minimises security risks from scenarios such as a student being accidentally given a staff role.

In order to ensure the automatic synchronisation process handles mismatches appropriately, there are number of different statuses a student can be assigned which determine how the process manages mismatches:

- Active *(default)*
- Forced active
- Left
- Forced left
- Auditing
- Withdrawn

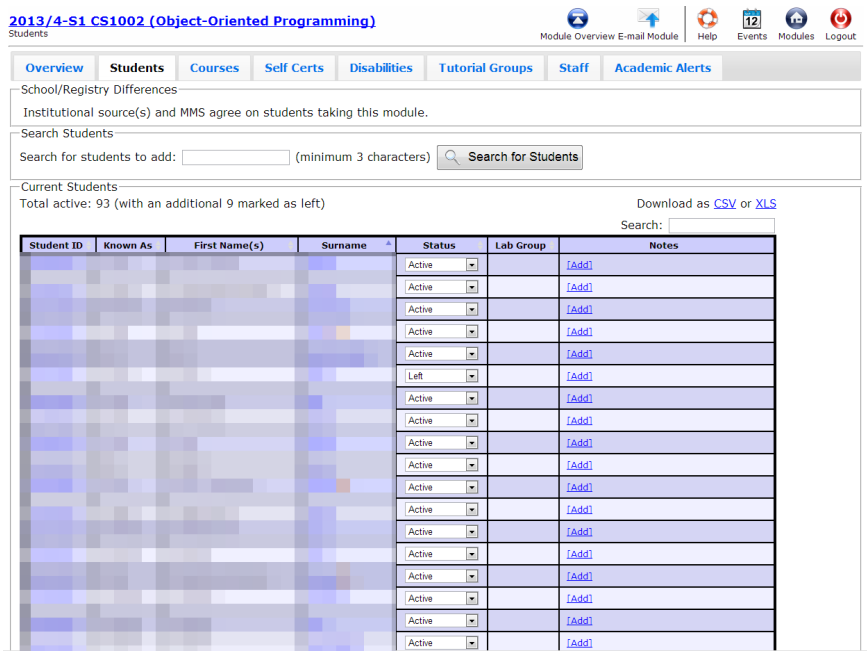


Figure 7.2: Screenshot of Configuring Enrolments on Module

"Active" and "Left" statuses simply represent that a student is enrolled on the module, and that a student has been on a module but has since been unenrolled, respectively. The "Forced Active" and "Forced Left" variants of these roles are used to tell the synchronisation process to ignore mismatches in central records which would otherwise change the student's status.

"Auditing" is a special case status where a student is sitting-in on a module, typically in addition to their normal studies, and as such they do not have a matching association in central records. Such students do not see assessed work, and are not returned as part of the results at the end of the module. Through explicitly supporting auditing roles MMS can ensure it accurately models module attendance without requiring workarounds to avoid problems in reporting.

Lastly "Withdrawn" is used where the student is still enrolled on the module itself, however has withdrawn (temporarily or permanently) from their degree course. Ideally this status would be modelled by an independent field for storing status of a student on their degree(s), however as an implementation compromise this status is used instead.

7.2.3 Staff Roles

Central records at St Andrews contain limited data about staff involved in modules, typically only recording the academic responsible for the module and in some cases lecturers. As such, automatic import of this detail from central records is of limited use, and MMS' own records are maintained independently.

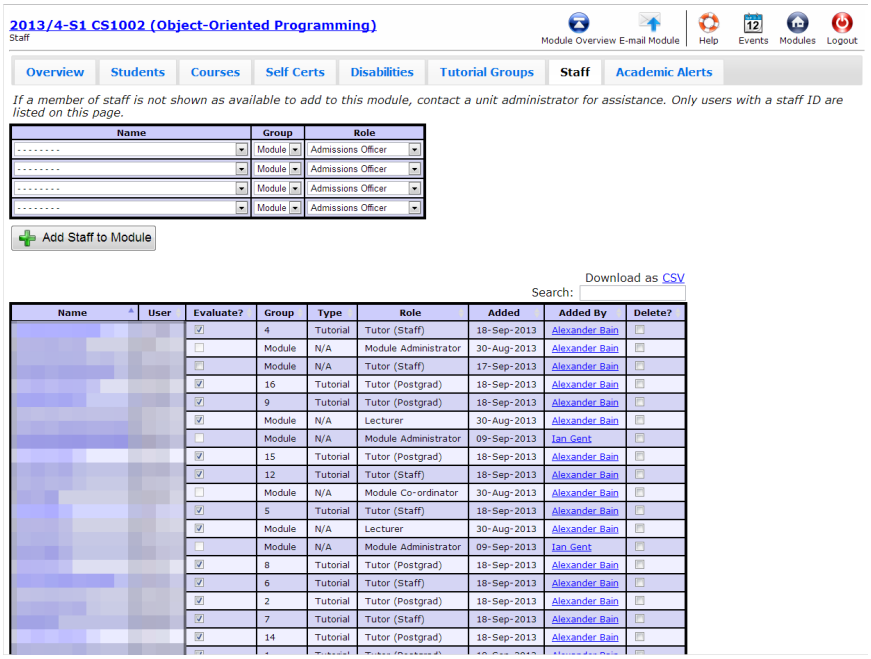


Figure 7.3: Screenshot of Configuring Staff on Module

The staff management tools consist of two parts; a list of existing staff on the module and/or groups within modules, and fields for adding staff roles to the module. The interface for adding staff roles differs from the student interface in two ways – suggested staff are provided based on the list of staff attached to the academic school the module belongs to, and a number of field sets are provided such that multiple roles can be added in a batch. This interface is shown in figure 7.3.

Suggesting relevant staff in this manner is intended to simplify finding staff within the staff directory. A search field is provided for the limited number of cases where staff are cross-assigned from other schools or departments.. Problems can arise in cases such as postgraduate students who are considered tutors but have not yet been formally made staff, as the security framework within MMS will not allow non-staff to be given staff roles. Currently a workaround is provided where an override is applied to their records in MMS

to mark them as “staff-like”. Clearly this is not an ideal solution, however no better option presents itself.

The high quality of staff data contained within MMS has led to this data being reused by the EvaSys module survey tool (see section 6.2.4), and again this is a good example of the design principle of exposing data to other systems.

7.2.4 School-level Bulk Administration

The interface for configuring modules individually can be frustratingly limited and time consuming for staff in contexts such as setting up large batches of modules at the start of a semester.

While the significant variability of processes and requirements in higher education use cases has been discussed earlier, there are general patterns to module setup. A typical academic school might have 3-4 standard patterns for modules, with a number of unique cases apart from those. These assessment patterns may be split based on degree type, attendance type, subject and other criteria as discussed in section 3.1. The ability to see an overview of all module configurations and readily identify exceptions follows with the design principles of providing tools to analyse data quality, and of pro-actively raising issues early.

The drawback to school-wide administrative tools is that in order to present a manageable amount of data, the interface must be narrowly focused on small sets of configuration options. This is a compromise which is suggested as practical in context as an alternative to the single-module configuration tools, however it is not generally intended as the sole interface for making these changes.

Coursework

Coursework within each school tends to follow a limited number of patterns within each school, depending on subject requirements in terms of assessment, and typically has consistent policies set on issues such as lateness and student anonymity during marking.

Coursework setup has clear benefits from school-level configuration tools. Consider a

school with a primarily essay-based assessment process (many art subjects follow this pattern). The school-level tools provide an overview of deadlines, submission windows (time between the assignment being available to submit work to, and the deadline), acceptable file types, assignment type (submitted on paper or electronically) and assignment labels. These details tend to be similar – for example modules with a one to two essays are likely to have similar due dates to ensure consistency in time available for students to complete their work. Schools generally have similar processes for coursework submission.

Final Grade Calculation

The policies for calculating final grades can be managed across an entire school in order to ensure consistency. This is particularly important in the case of schools which use customised rules, for example requirements that students pass all elements of coursework in order to be rewarded an overall pass in the module.

Module Evaluation (Survey)

Module evaluation is the last of the areas with a tendency towards highly consistent configuration. The school-wide interface allows mass setting of the member of staff who should receive evaluation reports; this can be the Director of Teaching, Head of School, a member of administrative staff, or another member of staff depending on the school.

This data is then exported to EvaSys as an XML document according to a schema defined by the EvaSys tool.

Staff

A mass-configuration interfaces are provided for managing staff attached to module, providing a table view with staff in rows and modules in columns. Role to display is chosen from a drop-down list at the top of the page, and assigned/revoked by selecting/unselecting checkboxes corresponding to each role/staff combination. For schools which decide teaching assignments outside of MMS initially, this interface is well suited to enabling administrative staff to mass-assign course coordinators, lecturers, etc.

A similar interface is presented for students, however it is read-only and used for verifying records and monitoring student numbers rather than configuration.

7.3 Student Management

7.3.1 Enrolment Tool

The INSIDE application allowed staff to request students confirm (or deny) their intention to attend a taught module. INSIDE could also present students with a number of questions as part of the enrolment process. That functionality had two key purposes:

1. Ensuring lists of students taking a module were accurate, considering the students as an original source for enrolment data and capturing that information to compare against central records.
2. Assisting with determining appropriate tutorial groups for students, by allowing students to indicate times when they could or could not attend tutorials.

There are a number of scenarios where central records of the modules which students are intending to take could be inaccurate. For example students attempting to change modules without following the correct processes to ensure their records were updated, less commonly students being advised onto the wrong modules, or their module choices being recorded incorrectly. This lead to a high risk of mismatches between central lists of students on a module (held by the Academic Registry), the school's understanding of which students were taking a module, and the students who thought they were listed as taking the module.

In this use case, INSIDE had also attempted to improve data management by providing tools to check central records against locally held records, and report those differences to Registry, however this required staff to be pro-active in looking for problems.

In MMS, similar functionality was provided by the Enrolment tool; screenshots of this tool are shown in figure 7.4. The Enrolment tool could be added easily to any module

in MMS, and questions and answers were user defined. These questions were multiple-choice rather than free text, such that student answers were constrained to a sensible set of possibilities. Questions could also be shared between modules, for example all modules in a single year group frequently have the same options for tutorial times. This sharing of common questions and answers minimises duplicate entry of data.

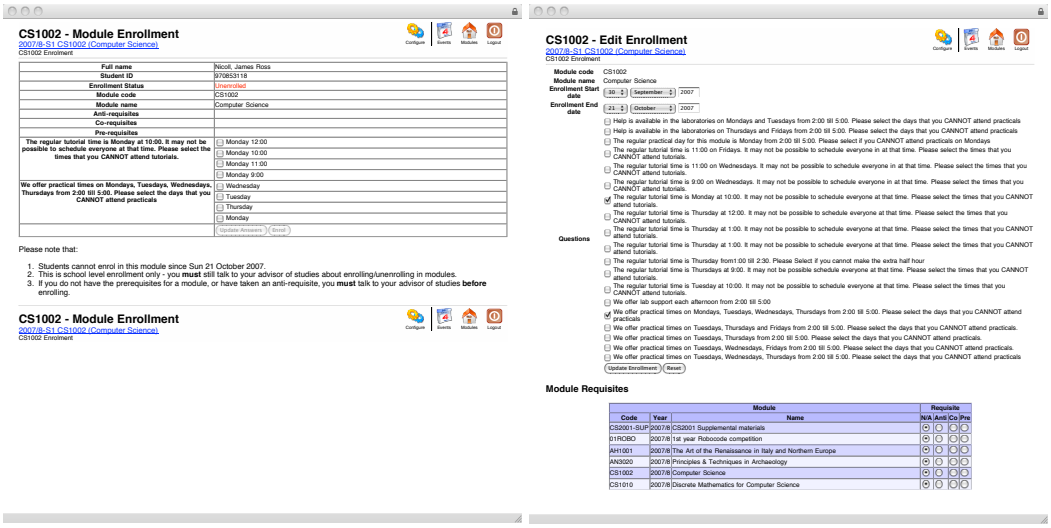


Figure 7.4: Screenshots of Enrolment Tool

In terms of question methodology, it is worth noting that the School of Computer Science asked when students could **not** attend a tutorial, rather than when they could. In their experience students would interpret the latter to indicate preference, and the former requirement.

The enrolment tool also provides a final sanity check of prerequisites, co-requisites and anti-requisites to the module, in case a student had accidentally been advised onto a module despite guidance to the contrary.

Improved reliability and currency of central data through increased visibility and workflow changes have somewhat mitigated the need for this tool in recent years. The tool is still however used in cases where tutorial group enrolment is staff-driven rather than student.

7.3.2 Group Signup Tool

In 2008 a number of schools, principally the schools of Economics and Finance and International Relations reported finding it challenging to manage the process of student signups

to tutorial groups. These schools had some of the largest individual taught modules, and were struggling with the logistics of the signup process; attempting first-come first served with 200-300 students and 2-3 administrative staff (or worse, a single lecturer) is obviously going to have issues involving workload on staff (or even just trying to get that many people to make an orderly queue). Random assignment to groups is unpopular and risks students being placed on groups they cannot attend (causing further disruption as they're moved, with potential knock-on effects as their groups may be left over or under-capacity).

The group signup tool was developed to deal with this requirement. Initial design and development was done by Stuart Purdie, with significant redevelopment done by myself to deal with issues under high load. The group signup tool allowed students to self-enrol onto groups (tutorial, lab or otherwise), with constraints imposed on maximum group size, last date on which students could move between group, last date that students could join any group, etc.

The tool proved extremely popular with both students and staff, excluding some early incidents where groups were accidentally allowed to become over-populated due to race conditions.

Contrast with the enrolment tool, designed to the requirements of Computer Science, who wanted to acquire information such as a student's availability on specific days/times, and then handle group assignment themselves. This required more staff time, but allowed groups to be designed with specific goals in mind (for example providing a mix of skill level).

7.3.3 Tutorial Attendance System

The Tutorial Attendance System (TAS) tracks students' attendance at tutorials. The tutorial attendance system was originally developed by Bin Ling, and later redesigned by myself, although the user interface has remained mostly as-is. The original version also didn't use actual dates as the labelling on tutorials, as it was considered desirable to have the flexibility of free-form labels, TAS has since changed to use actual dates to ensure data integrity. The tool has also been retroactively modified to support auditing of both configuration changes and attendance recording. Sample screenshots are shown in 7.5.

7.3.4 Lecture Attendance Management

The Lecture Attendance Management (LAM) tool provides tracking of lecture times, and student's attendance at those lectures. While there are similarities to the tutorial attendance tool in design and function, the tool differs in a number of key areas:

- It tracks attendance at individual, independent events, instead of across sets of events to which students are expected to attend only one occurrence of.
- The user interface has been re-designed to better support display of a very large number of attendance data points.
- It allows students to record their own attendance, rather than requiring it to be entered by staff.

The requirements for the tutorial attendance tool were that it would track sets of events, and students would attend just one event within each set. This was a good fit for tutorials, where the students on a module were split into groups and each group had a specific tutorial to attend out of all tutorials running. These event sets were recorded as date ranges (typically reflecting the Monday and Friday of a week), rather than as a specific date and time.

However lectures generally run several times a week with students expected to attend all lectures (there are exceptions to this, in the case of extremely heavily subscribed modules, however these are very rare). These lectures have a specific start and end-time, as well as location, and this detail is required for allowing students to record their own attendance.

There are also limitations in the user interface of the tutorial attendance tool, with regards to handling frequent events, in that the columns are relatively wide. This means that using the same user interface for lectures would result in impractically wide pages (substantially wider than most displays, likely a multiple of the screen width). The interface was therefore "slimmed down" for the lecture attendance tool, removing detail and using abbreviations to minimise the screen estate required for each data point.

The last requirement, of allowing students to record their own attendance, is the biggest change. To support this dedicated PCs were set up in some lecture theatres, with magnetic strip readers attached. On entry to the lecture theatre students would swipe their university

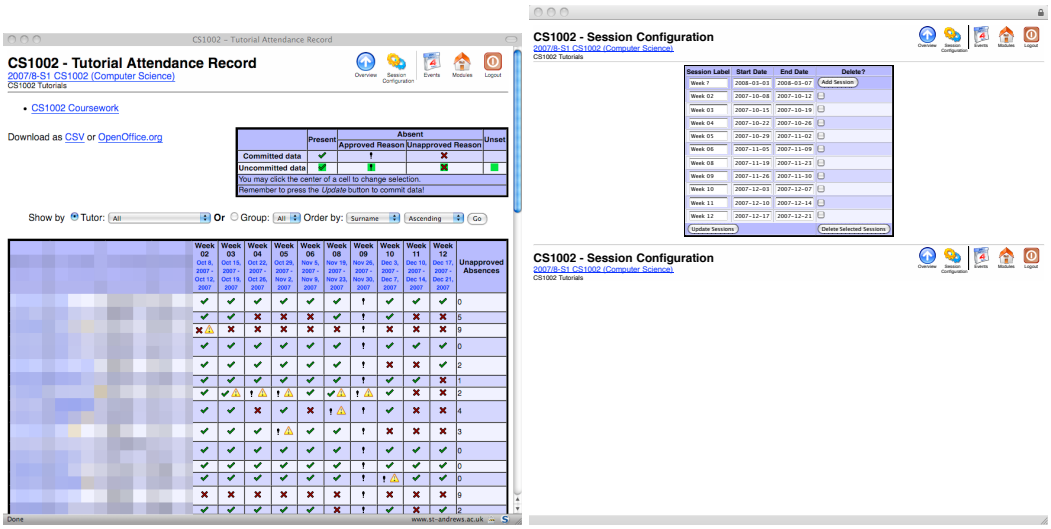


Figure 7.5: Screenshots of Tutorial Attendance System

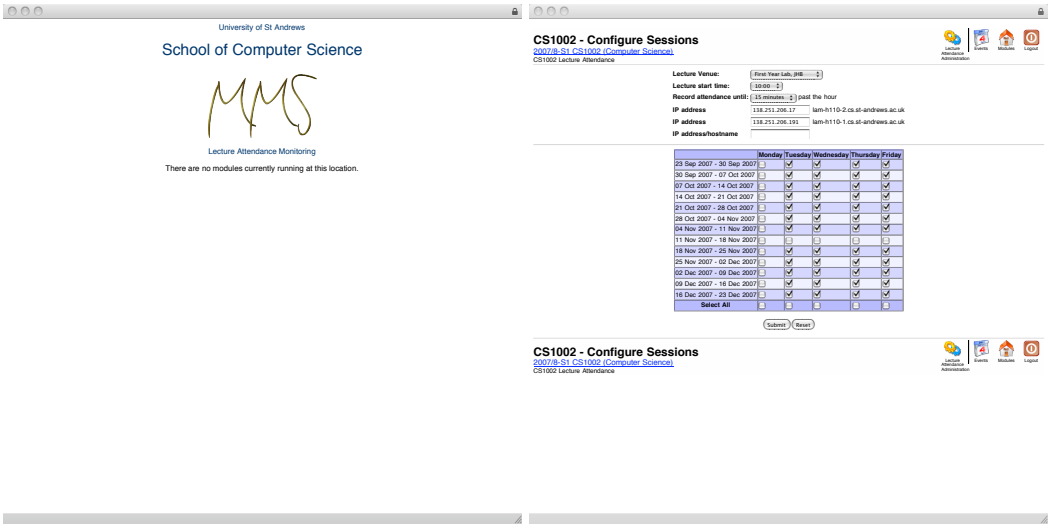


Figure 7.6: Screenshots of Lecture Attendance Management

ID cards through the reader on one of these PCs, and the card number would be sent to MMS, which would record the time, card number and IP address of the client PC.

The time and IP address was compared against a list of currently running events, and where a match was found the student was recorded as in attendance.

Given that the tutorial and lecture attendance tools do provide similar functionality, it would seem obvious to merge the two tools' code and data structures. This would in theory reduce the number of different tools users have to be trained in using, and also reduce the code required. In doing so, this limits the scope for bugs in the code, and ensures that any improvement in one tool applies to both.

Unfortunately, to do this would not only require a common user interface to be developed that encapsulates the needs of both systems, but also that existing data from both tools converted to common format. Developing a common format for both systems has so far proven challenging due to the differences in details on events, and ultimately a lack of any strict requirement to merge the two tools has meant this work has not yet been tackled

7.3.5 Announcements

While e-mail is a straightforward way of delivering notifications of events, problems, changes, etc. it depends on the audience deliberately checking their e-mail. This can make it a poor choice for delivering time sensitive communications, for example notifications of change of venue, cancellation of lectures, etc. The School of Mathematics and Statistics were the first to suggest using MMS to deliver announcements to students. The first version of this worked by attaching announcements to modules, and all announcements of relevance to a student were displayed on login.

This approach proved overly limited in that it simply changed the nature of the problem – now students are required to log in to MMS instead of e-mail. A better approach would be to automatically e-mail a copy of a message to students, although even that still would require action to be taken on the part of the recipient. A longer term solution might be to send such notifications to mobile devices (see also section 12.5.4), although there are a wide number of other communication technologies in use in higher education [120] which could be considered.

7.4 Student Assessment

MMS provides two tools used in assessment; a coursework tool intended for submission and marking of a number of independent items of work, and an exam tool for tracking marks from exam(s) taken as part of a single diet. These tools are intended to capture data as close to original source as possible, for example coursework directly from students and marks from academics. Some schools insist on mark entry being done by administrative staff from spreadsheets or written notes provided by academics.

7.4.1 Coursework Management

Virtually all modules at St Andrews have some form of assessed coursework, with only a tiny fraction limiting assessment to end of term exams. In the academic year 2012/3 alone, there were over 4,500 coursework assignments set for students. Clearly streamlining management of coursework submission and marking is a significant opportunity for reducing overheads.

MMS provides a single multi-role tool for coursework submission, mark tracking and feedback. It has evolved significantly since its initial development as replacement for the Document Approval Tool (DAT) from TAGS.

Online Assessment Tracking System

The first version of the coursework tool was the Online Assessment Tracking System (OATS). OATS provided facilities for students to upload coursework either to predefined assignments, or to "ad-hoc assignments". Uploaded work could then be approved or rejected by staff. Marks and/or written feedback could be set on the submission for the student to view.

Configuration of predefined coursework assignments required details such as assignment title, due dates (by which the student was expected to have submitted their work) and acknowledgement dates, by which staff were expected to have marked the coursework and/or provided feedback. Ad-hoc assignments were intended for use in exceptional cases such as submitting work in progress, adding supporting documentation to coursework after it has

been submitted, where configuration problems mean work cannot be submitted normally, or in any other cases where a predefined assignment was not a suitable answer.

Unfortunately ad-hoc assignments proved to be a significant cause of confusion, in particular in best practises in using them. There were a number of occasions on which students misunderstood the interface, and uploaded work into the ad-hoc assignments despite the availability of a more appropriate predefined assignment. Further, no functionality had been provided to transfer coursework from these ad-hoc assignments into regular assignments, which made recovery from incorrect usage time consuming.

The accept/reject mechanism was used to indicate whether the work was as-expected, for example work might be rejected if the wrong file or file type was uploaded, but might also be used if coursework is well below the standard expected. Students could also attach comments on uploaded files for staff to read, examples of where this is useful include providing a note on problems they encountered, any special instructions on how to compile a program, etc.)

In comparison to TAGS, OATS introduced the automatic application of lateness penalties in cases where work was submitted after the due date of an assignment, although it was limited to the lateness policy in use for the School of Computer Science. Assignment due and/or acknowledgement dates could be overridden on a per-student basis, for example where illness required a student was given an extension or even exemption from an assignment.

Following the MMS design principle of actively reaching out to users rather than passively waiting for them in case of problems, OATS could send e-mails to remind students and staff of work to be done. In the case of students this was limited to coursework due, and was only triggered after the deadline was missed, as it was felt that the system should not act as a substitute for organisational skills expected of students. For staff there were options to receive e-mail on students uploading work, submission deadlines being missed, and/or acknowledgement deadlines being missed.

Graded Online Assessment Tracking System

The Graded Online Assessment Tracking System (GOATS) was developed in response to changing requirements in terms of coursework marking processes. Screenshots of this revised tool are shown in figure 7.7. This revised tool incorporated a number of improvements

and functionality changes:

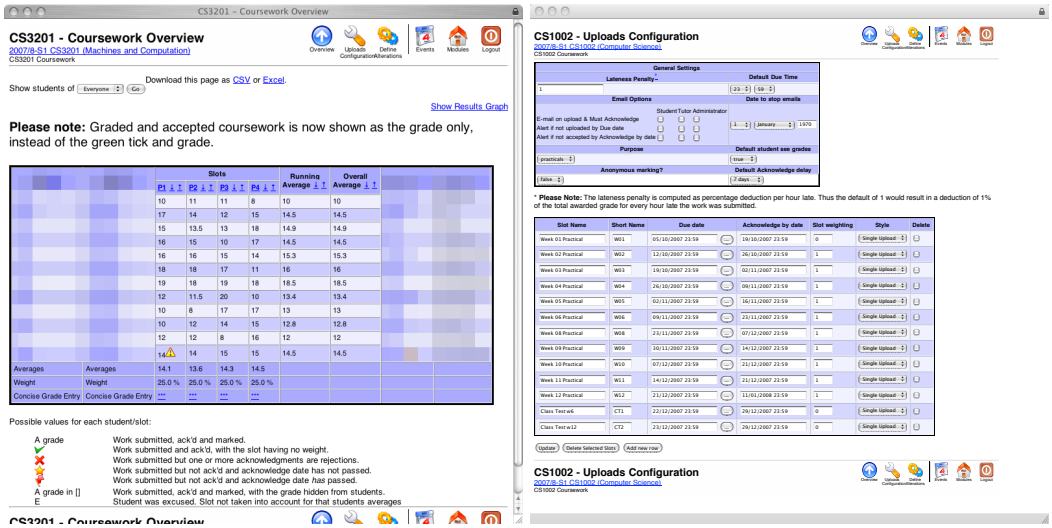


Figure 7.7: Screenshots of Coursework Management (GOATS)

- Marks were initially limited to the standardised St Andrews grading scale. This change was made in reaction to changes to marking policy across the institution, which meant that all work was expected to be marked against this revised grade scale. Later this would be revised to support other marking scales (see section 5.11.3).

- An API for extracting marks from a tool in an implementation agnostic manner had been added to the MMS core framework, and the GOATS tool implemented this API. This allowed other tools, such as the final grade calculator, to access marks in the coursework tool without requiring an understanding of its internal structure.

- Ad-hoc assignment functionality was removed in order to simplify the user interface. Ad-hoc assignments were rarely used correctly, had no strong requirement from users, and frequently caused additional work due to misuse.

- Staff and students could now attach uploaded files to feedback/comments on coursework. For staff this was intended primarily to allow the return of coursework with comments annotated on the work itself, whereas for students this assisted with the replacement of the ad-hoc assignments by allowing them to attach supporting files directly to the submitted work.

- Support was added for anonymous marking of coursework (see section 7.6.4 for further discussion on anonymous marking policies).

These changing requirements in terms of marking schemes later lead to development of the mark abstraction layer to manage the complexity of tracking and manipulating multiple different marking schemes. Ad-hoc assignment functionality was partially replaced by the file sharing tool, which was extended to provide private file space on a per-student basis.

Coversheet Generation

One of the frequently raised issues faced by academic staff with electronic coursework was the amount of time spent printing out coursework, and the difficulties in tracking coursework where coversheets were missing or incorrect. Coversheets are needed to ensure essays are associated with the correct student when marks are being recorded and feedback returned.

A combined solution for managing coursework coversheets and printing of coursework was proposed with the intent of reducing the number of independent documents that needed to be printed, and modifying those documents at the same time. The first version of this functionality required students to submit work in PDF format, but could then collate all documents within a single assignment into a single very large document, inserting coversheets between them to identify which piece of work belonged to which student. The resulting document could then be printed in a single batch by secretarial staff, stapled and passed along to markers.

A later version attempted to do the same for Office Open XML[1] formats (commonly referred to as the "Office 2007 formats", referring to the first version of Microsoft Office that made use of them). Although partly successful, the complexity of the Office Open XML document format made this substantially trickier and less reliable than for PDF documents.

Essay Conversion to eBook

Another approach considered was to eliminate the printing step entirely. The rationale most frequently given for printing coursework instead of marking on a computer was that reading on a monitor is harder than from paper, and this is generally supported by the

[1] `http://www.ecma-international.org/publications/standards/Ecma-376.htm` - accessed 12th November 2013

literature (O'Hara[96] and Shaw[119] are recommended articles on reading on screens vs. paper, especially in context of marking work).

Given that electronic paper devices such as Amazon Kindle[2], Sony e-Readers[3], etc. are marketed on the basis of being easier to read than a CRT, LCD or other computer display, it was hoped those devices might be an acceptable compromise for academics – not requiring printing, while being easier to mark than onscreen.

However, this required either that students submitted work in a compatible format (ePub[4], Mobipocket[5], or similar), or that documents were converted to such a format. Note that Mobipocket is the standard format for the Amazon Kindle e-reader, and ePub is used for most other e-readers. For a small-scale test, it made more sense to look at conversion of documents initially, rather than disrupting student assessments. The ePub format was chosen for this test as it was simpler to produce documents in from a server application – all supported methods of generation Mobipocket files use an external application to package the content[82].

The coversheet and document merging functionality developed provided much of the core of a document converter, allowing essays to be re-written into HTML while maintaining basic formatting (bold, italic and underline) for the first version. Once a suitable HTML document was generated from the source, packaging it with metadata to produce an ePub format e-book was relatively straightforward.

Tests with the conversion produced acceptable results – complex documents could cause problems when being converted to HTML, or lost too much formatting for the result to make sense, however most essays tested were simple enough for the result to be clear (if rather unimaginatively formatted).

A small number of academics were shown these essays or given the ability to use the conversion functionality in MMS themselves. Feedback was disappointing, most found the premise interesting, but display size, lack of ability to mark up work, and time spent managing documents (to transfer to e-readers) were all significant complaints. Potential future developments in this area are discussed in 12.5.2.

[2] `https://kindle.amazon.com/`
[3] `http://www.sony.co.uk/product/rd-reader-ebook/tab/overview`
[4] `http://idpf.org/epub`
[5] `http://www.mobipocket.com/`

7.4.2 Exam Marking Tool

Marking of end of semester exams is a complex and intensive process, where numerous academics work on shared data sets against an aggressive time schedule. It is common at St Andrews for marking of exam papers to be shared by several members of staff, in some cases with different academics marking different questions. This poses challenges in collating the marks in an accurate and timely manner once prepared. There is a requirement for confidentiality in handling of marks, which means many easy options for sharing files such as e-mail or Dropbox may be unsuitable.

Clearly, there is a need for a tool which can collect these marks, and automate the process of collating them. The coursework tool had several key limitations that prevented it from being used for marking end of semester exams:

- Configuration and mark entry interfaces are orientated towards managing a single assignment at a time. Each exam question would require a separate assignment entry, meaning the configuration and marking would be impractically time consuming due to need to constantly change between pages during the processes.
- No provision was made for optional assessments within sets. Exams frequently have sets of questions from which the student picks a subset to answer, and the coursework tool could not support this.
- There was no support for transforming marking scales. Exams are typically marked on a linear scale, but may then have marks transformed to a different scale (for example the 20-point grade scale used at St Andrews).

Note that these constraints reflect the coursework tool in 2005 when the exam tool was designed, support for optional assessments and mark transformation have been since added to the coursework tool.

The exam tool (shown in figure 7.8) provided a single point of truth for student's exam marks, and was the first tool to audit changes to marks entered. This audit trail was key to encouraging confidence in the correctness of the tool, enabling staff to quickly determine the origin of each mark entered. Staff could work independently and asynchronously, with user input managed as a set of changes to be applied. This meant the staff responsible for

marking different sections of an exam could work concurrently on the same exam, without risk of accidentally overwriting other users' changes. This is particularly important as it is common for multiple staff to be marking different exam scripts or sections of the same exam script in parallel.

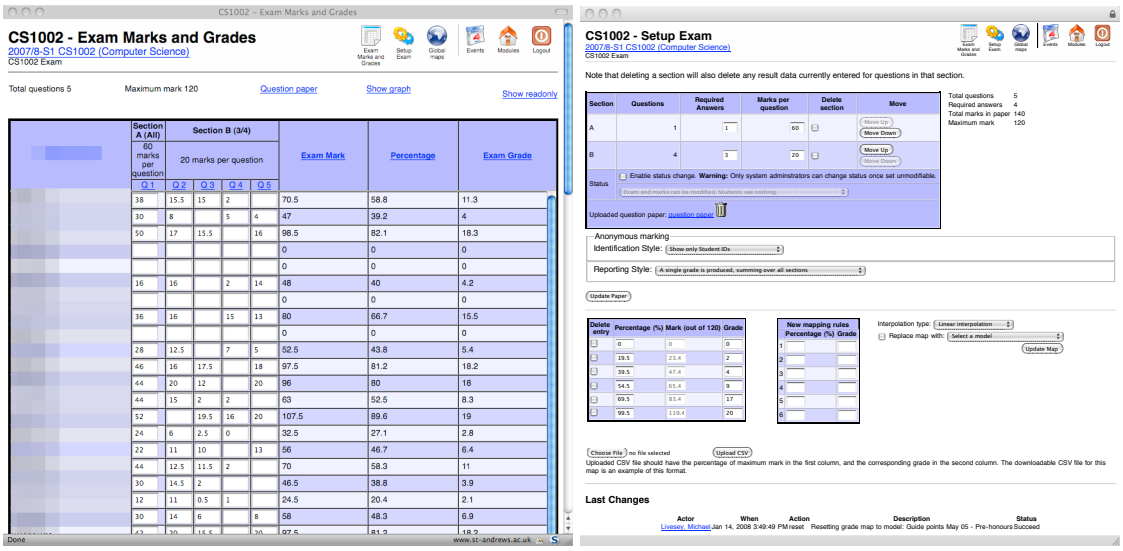


Figure 7.8: Screenshots of Exam Mark Repository

The interface for entering marks was designed around a spreadsheet-like grid for, with individual questions represented as columns and students as rows. Questions were grouped in the interface by the exam section they belonged to. Exam sections in this context contain are sets of one or more questions, of which students are expected to answer a defined number. All questions within a single section are worth the same number of marks. For example a section may contain 3 questions each worth 20 points, of which students answer 2 questions for a maximum of 40 points available from that section.

Once marks have been entered, the marks from each section are then summed to give a student's total mark for the paper. If a student answered more questions within a section than required, the highest marks from the section were used up to the expected number of individual marks.

Mark Conversion

St Andrews requires grades on a 20 point scale for reporting purposes, and as such exams typically required marks to be translated (or "mapped") into grades. The mark conversion

tools described in 5.11.5 were initially developed solely for the exam tool. Later adoption of MMS to all academic schools introduced a requirement for accepting coursework marks on varying scales, which would then be converted before reporting. As such, the conversion process was extracted from the exam tool and generalised to allow other tools to utilise it.

Mark Reporting

As with the coursework tool, the exam tool implements an abstract interface for exposing grade data, allowing it to be used as a data source for the final grade tool. This interface is described in 5.13.

7.5 Grade Reporting

Having collected students' coursework and exam assessment marks, it is logical to make further use of this data within MMS, following the design principles of acquiring data from as close to the original source as possible (in this case academics via MMS) and exposing data as widely as possible.

The final grade tool (shown in figure 7.9) collates data from other tools in MMS using the abstract interfaces provided by the framework (see 5.13.1) and produces a result for reporting to central records.

The initial design was based on workflows in use in the School of Computer Science, where marks for each assessment type (coursework and exams) were collated independently before being used to calculate the final mark. These component marks were typically used to calculate a weighted mean (for example 60% coursework, 40% exam) as the final mark.

Comparisons to the gradesheets in Moodle, WebCT and other VLEs are inevitable, however there a number of key design differences between these, most notably that the tool in MMS uses scripted calculation models rather than conventional spreadsheet-style formulae. Most importantly, models **must** include a unit test suite as part of their definition, which requires staff to consider suitable tests and helps ensure correctness of the implementation.

The intention of the tool is also subtly different; the final grade tool by default displays

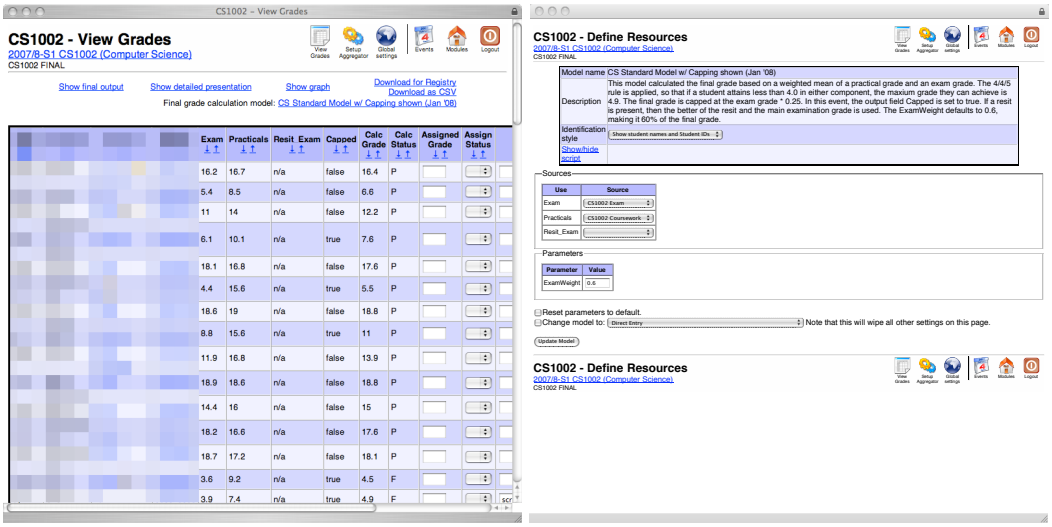


Figure 7.9: Screenshots of Final Grade Calculator

a high-level overview of a student's attainment across a module, only providing a detailed breakdown (such as individual coursework items) as an alternative view, whereas gradesheets focus on displaying all marks across a module at once.

The calculation models are based on ECMAScript, enabling straightforward configuration of relatively complex calculation models. These models can include a number of features which would be challenging or impossible to implement in other systems which lack support for conditionals in grade calculations, such as Moodle[84]. Consider the scenario where a bonus mark is awarded if a student achieves a grade 17 or above in all of their coursework; this is trivial to express in a language with support for conditionals, but nearly impossible otherwise (the same result can be emulated using division, rounding and min/max, however it results in an overly-complex and difficult to understand formula).

Further details of the final grade scripting environment and its history are discussed in the academic policy section 7.6.1.

7.5.1 Model Tests

The last key difference compared to similar tools is the provision of a test suite to allow end-users to provide integration tests for their rules. This follows the design principle of pro-actively raising issues early, by catching implementation issues at implementation time

instead of during usage. This is considered important for assuring quality and correctness of the results, especially given that these results determine whether students pass their degrees.

Calculated grades could then be manually modified if staff feel that a student deserves a higher/lower grade than has been calculated, possibly due to exceptional circumstances or based on adjustments required by an external examiner. Result status (pass, fail, deferred, etc.) is also automatically calculated from the final grade, but can also be overridden by staff if needed.

Once all grades and status are confirmed by an exam board, they can be exported in a file format suitable for uploading to SITS:Vision by the Academic Registry.

7.6 Academic Policy

A number of tools within MMS use server-side scripting (see also section 5.12) to support development of academic policies. This was initially developed as part of the final grade tool, and background is provided as part of the description of that tool's functionality, below.

7.6.1 Final Grades

One of the functions of the final grade tool is to calculate grades for students based on coursework and/or exam grades as appropriate. This final grade is typically calculated using a weighted average of the source grades, as an example the module CS1002 currently has a model described as "2-hour Written Examination = 60%, Coursework = 40%", which is calculated as:

$$grade = (coursework \times 40 + exam \times 60) \div 100$$

More complex models may have conditional elements, multiple outputs (for example warnings in case of unusually distributed input values), etc.

Rule Chains

The first version of the final grade tool used rule chains to describe the operations to be performed on grades. This was intended to provide a way of specifying the calculations without requiring any knowledge of programming. Rule chains consisted of a limited set of operations and were intentionally not Turing-complete (no support for loops). These operations were:

- minimum
- maximum
- add
- subtract
- multiply
- divide
- if-greater-than
- if-less-than

Most of these (all but the conditional operations) operated on two mark parameters and produced a single result. For example, to calculate the result for a 40/60 split of coursework to exam, there might be three rule chains. Two of the chains would be used to multiply the coursework and exam grades by their respective weights (40 and 60), and would then feed into the third chain. That chain would add the results of its source rule chains and then divide by 100 to produce a final grade.

This model proved a poor fit, with staff requiring training to make effective use of the design (in comparison to formulae based solutions), while failing to have the readability of solutions such as scripting. Actual models used also were significantly more complex than anticipated, and this approach was abandoned for a solution using server-side scripting.

ECMAScript

ECMAScript[6] (more commonly referred to as Javascript) was chosen as a scripting engine as it was readily available, and the resulting scripts would generally closely resemble mathematical formula. This scripting engine is based on the Mozilla Foundation's Rhino project[7]). Continuing with the earlier example of a 40/60 coursework/exam weighted mean, the calculation might be scripted as below:

```
(Coursework * 40.0) + (Exam * 60.0) / 100.0;
```

Note that the explicit precision is important due to use of floating point numbers. More complex examples are also possible – for example Psychology students may be given a "bonus" mark if they received high marks on their coursework. As a script this would appear as:

```
var bonus;

// Calculate bonus mark based on coursework component marks
if (Coursework1 >= 17.0 && Coursework2 >= 17.0) {
    bonus = 1.0;
} else {
    bonus = 0.0;
}

var Coursework = (Coursework1 + Coursework2) / 2.0 + bonus;

(Coursework * 40.0) + (Exam * 60.0) / 100.0;
```

The scripting environment evaluates the script to find the result, and therefore there is no explicit return of the final value. Clearly this style of model would be challenging or impossible to implement without support for conditionals. There is a significant benefit to scripted models in comparison to spreadsheets in terms of readability, especially given the ability to include inline comments.

[6] http://www.ecma-international.org/publications/standards/Ecma-262.htm - accessed 19th October 2013

[7] http://www.mozilla.org/rhino/ - accessed 19th February 2008

7.6.2 Coursework Grades

After the success with the final grade calculation scripts it was logical to extend this functionality to other grade calculations. The resulting grade for most coursework is a simple weighted mean of the awarded marks. Some modules use a model of "Any n pieces out of m", but that too is readily modelled by having a field to determine number of pieces of work expected.

More complex models such as "Two short essays or one long essay" however were not anticipated during initial implementation, and through use of server-side scripting such models can be readily added post-deployment. In extreme cases models such as the one below (used in Chemistry) can be required, and this sort of complexity was never anticipated during initial design and further illustrates the requirement for server-side scripting to support unexpected cases. Here A to H represent coursework marks, f is a value between 0 and 1 inclusive representing the students' contribution to the group work components G and H:

$$2A + B + C + \frac{2(D+E+F)}{3} + 2f(G+H)$$

This is difficult to model as weighted elements, but relatively straight-forward in ECMAScript.

7.6.3 Double Marking

Double marking is used as a method for improving the reliability of marking[138], but adds complexity to the marking process in requiring two copies of an assessed piece of work to be disseminated, and marks to be collated and agreed upon after the marking process has been completed. MMS automates much of this – for electronically submitted work it makes it trivial to deliver individual copies to each marker, and of course it can manage independent tracking of marks as well as mismatch handling.

Double marking is a clear example of where shared data sources and workflow tools can streamlining processes. Sets of marks are entered into MMS independently by markers, and reconciliation is then performed by highlighting where marks do not match, and staff collaborating to decide a final outcome in such cases. Future work could allow for other

forms of reconciliation in case of differences, for example mark averaging, use of more than two markers, etc.

7.6.4 Anonymous/Blind Marking

The value of anonymous marking (sometimes referred to as "blind marking") remains a matter of debate[15][16][89][99][11], however the NUS continues to campaign for anonymous marking[95]. Ultimately, teaching staff provide demand for the functionality, and MMS was well placed to mitigate some of the challenges in anonymous marking.

There are two key criticisms of anonymous marking. Firstly, that it can be difficult to implement, with increased risks of coursework becoming dissociated from the student, or returned to the wrong student[8]. Secondly, that feedback cannot be tailored to individual student's, resulting in an impersonal feeling to the process[135].

Regrettably, the latter is an inherent result of the anonymisation process, however it is hopefully clear that a tool for managing data flow is well placed for help manage the former. MMS was given functionality to anonymise students in views of marks, using the student's matriculation number (a nine digit unique ID) to identify them instead of their name. This decision is an institutional policy intended to enable identification of students, for example where they may require assistance with a specific area, while avoiding showing trivially identifiable names.

There have been predictable issues with student's having identifiable writing styles, or simply ignoring instructions and writing their name on their work. Marks are associated within MMS' records to the student as they are entered, and accordingly there is no significant risk of work or marks to become disassociated from the student who wrote it.

Automatic coversheet generation has been added to minimise risk of students either failing to identify themselves at all on work, or providing excess details (i.e. their full name). Staff have also complained that MMS' own internal user IDs are exposed in URL parameter, making it possible to identify students by noting the ID and using it to construct a URL for the user details page.

There is further discussion of anonymous marking in the future work chapter (see 12.5.1).

7.6.5 Lateness Policies

Lateness policies at St Andrews take a number of forms, from a proportional loss of marks over time, to a complex stepped penalty system with minimums applied based on awarded mark. Although load during submission time is sometimes a factor in deciding lateness policies[124], performance of MMS has been sufficient that this has not been an issue, although it is common at St Andrews for lateness policies to allow an hour's tolerance between published submission deadline and actually applying penalties. Stoneham's work[123] is of potential interest when considering lateness policies in the context of electronic coursework submission.

Lateness policies had been initially implemented as Java code as part of MMS' core, and were chosen from a list by users, however this model meant that any new lateness policies, or any corrections to policies, required not only developer time to implement and test, but were then delayed until the next release of MMS.

The use of compiled models additionally meant that users had no practical way of inspecting the model's source code, meaning there was a risk of subtle errors. This was compounded by a tendency for staff to provide requirement changes as short written changes to an existing requirement, rather than as a revised complete specification, providing additional scope for complexity and ambiguity.

Server-side scripting for final grade calculation (as discussed above) was positively received by most schools, this approach was adapted to provide scripted of lateness policies. A Javascript API was developed that provided the original mark for a piece of work, and a breakdown of lateness in both hours, days, weekdays (Mon-Fri) and weekday hours. This hid a lot of the complexity such as GMT/BST timezone shifts from the end-user, while giving them the power to write their own bespoke scripts.

Unit testing tools were also provided along with the scripting environment, allowing staff to enter hypothetical marks, due dates and submission dates to build test cases used to verify the result of a calculation.

The result was a reduced turn-around on changes from 2-4 weeks, to an hour or less depending on the model's complexity. It also improved accuracy by allowing staff to inspect and understand the process, providing a "many eyeballs"[111] approach to managing com-

plexity and improving model validation.

7.7 Learning and Teaching

One of the major drivers for change in 2010 was the university's move from WebCT. It had initially been hoped that MMS could be used as the institutional VLE, and to fulfil this goal we looked at extending the file share and notebook tools to provide content management and forum functionality, as well as adding a quiz tool.

However, while these two tools brought in much of the functionality expected from a VLE, ultimately there were still elements (discussion forums, quizzes) that were under-developed, while others were still absent (Wiki, database of terms, prepackaged content, etc.)

As a result, the migration from WebCT was performed to a combination of MMS as a primarily-administrative tool, and Moodle as a VLE. The migration itself was done by the same team that worked on MMS, with the support of number of postgraduate students, however the specifics of the migration process are beyond the scope of this thesis.

7.7.1 File Share Tool

The file share tool is a simple shared file space, which can be configured in a number of ways, including:

write-only A coursework drop box for students to submit work to.

read-only A repository of course materials for students to download.

wead-write A space for students to share collaborative group work.

No support for more complex file permissions was implemented, such as allowing students to see only files they have uploaded, or allowing students to only see files they or other students in the same tutorial group, have uploaded. A screenshot of this tool is shown in figure 7.10

Files are stored on disk in a structure matching that of the file share tool provided by TAGS; a single directory per file share tool, with two files for each "real" file uploaded. This file pair consists of the stored file itself, a second file containing metadata such as the unique ID of the user who uploaded the file.

Figure 7.10: Screenshot of Scratchspace Tool

"Scratchspace" Tool

When originally ported to MMS, the coursework tool had a feature (ad-hoc assignments) which allowed students to upload files as submitted work without requiring an assignment to be defined ahead of time by staff (discussed in detail in 7.4.1).

As usage of MMS expanded, it became clear that users were finding this confusing; work would be accidentally uploaded to ad-hoc assignments despite a correct assignment being available, or would upload multiple versions of the same piece of work into different assignments.

The "Scratchspace" tool was a short-lived attempt to blend the file share and coursework tools, providing a space for students to upload files without requiring any specific structure. This would both replace these ad-hoc assignments, and provide an online secure space for students to store files they were working on as part of their studies. It derived much of its core from the file share tool, while adding significantly more powerful configuration options, enabling it to be used as a drop-box (where students could write but not read), or

for file sharing within a group of students, amongst other scenarios.

It was developed independently of the file share tool rather than replacing the existing tool, in the expectation that there was demand for providing both tools as individual options. Staff had frequently complained contradictorily both of the rate of change, and of the lack of change, and in providing an improved tool option without requiring users to change to it, it was hoped to find an effective compromise position.

Content Tool

The "Scratchspace" file sharing and storage tool was adequate for simple use-cases, however it lacked the fine-grain permission controls, content layout, and other functionality that was expected for content management. As a prerequisite of the move of content from WebCT, it underwent an almost complete rewrite, resulting in a new software architecture for the tool, as well as significant changes to its underlying data structures, and of course user interface.

The result was a massively improved tool, which provided a streamlined interface while adding in content metadata and more flexible content types (folders, external URLs, etc.) It did not cover quite the same extensive range of content handling options as WebCT or Moodle, however it was not intended to either; the design was to cover most cases, and to do so simply. A screenshot of the interface is shown in figure 7.11.

The tool utilised the FileAPI[8], HTML 5[9] **drag and drop** functionality and XMLHttpRequest level 2[10] to enable users to drag files from outside their browser directly into the page to be uploaded. Drag and drop was a major part of the revised user interface, and the tool was one of the earliest production uses of the Javascript File API (many drag and drop file upload interfaces at this point in time were Flash based), in part due to support for the feature being limited to the Firefox browser. Compared to the previous process of using a file selector dialogue, this interface was anticipated to be easier and faster, and informal feedback from users suggested this was the case.

[8] `http://www.w3.org/TR/FileAPI/` – accessed 4th May 2013

[9] `http://www.w3.org/TR/2013/CR-html5-20130806/` – accessed 20th October 2013

[10] `http://www.w3.org/TR/2012/WD-XMLHttpRequest-20120117/` – accessed 20th October 2013

The extremely early adoption of a number of standards caused a number of issues, however. Some of the implementation had to be re-written as changes were made to both the FileAPI and XMLHttpRequest level 2 specifications. Users reported confusion caused by lack of widespread browser support for the functionality (requests to adapt it to work in browsers without suitable specification support were common). Experiences gained developing this functionality were later used to assist with the Moodle drag and drop plugin[11], written by Davo Smith, specifically in relation to handling differences in specification implementations between the Chrome and Firefox browsers.

7.7.2 Notebook Tool

The notebook is a very simple message board, ideal for allowing students to discuss ideas, although rather limited by the lack of support for more than one message thread. A screenshot of this tool is shown in figure 7.12. The notebook tool was initially a direct replacement for the equivalent tool from TAGS, including being data file compatible.

The original tool had stored messages in a pair of plain text files on disk (one pair for each tool created). One file is a quick reference which holds the serial number to be assigned to the next notebook entry (message), while the other holds the notebook entries themselves, along with metadata such as who wrote the entry and when it was written. The entries themselves were encoded into HTML before being written to disk, allowing multi-line messages to be stored on a single line through replacement of line feed characters with the line break HTML element "
". An example of this format is shown below:

```
26
1140443198
288
Question 1: How do ConnectionExceptions and NoRouteToHostExceptions differ?
27
1140443886
194
Some how my original post was deleted, so here is it again:<br/><br/><br/><br/>
28
1140445331
267
Where is the Authenticator Class used in the Sun provided JDK or its extensions?
```

[11] https://moodle.org/mod/forum/discuss.php?d=181988 - accessed 20th October 2013

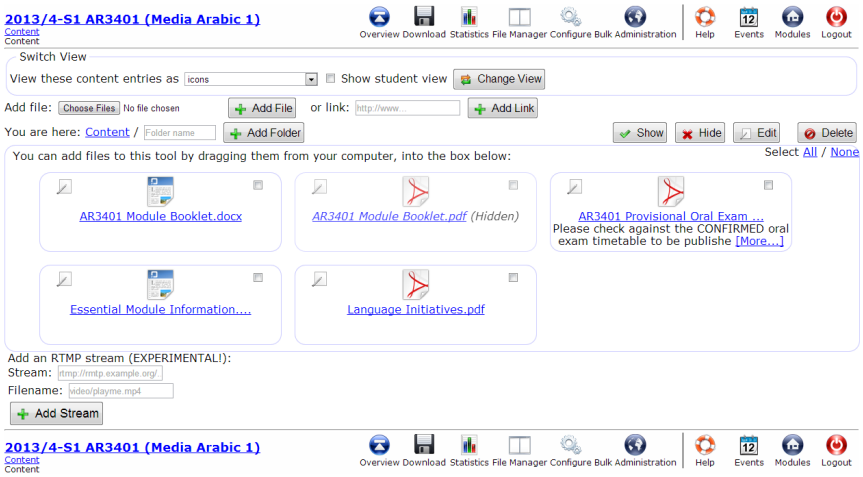


Figure 7.11: Screenshot of Content Tool

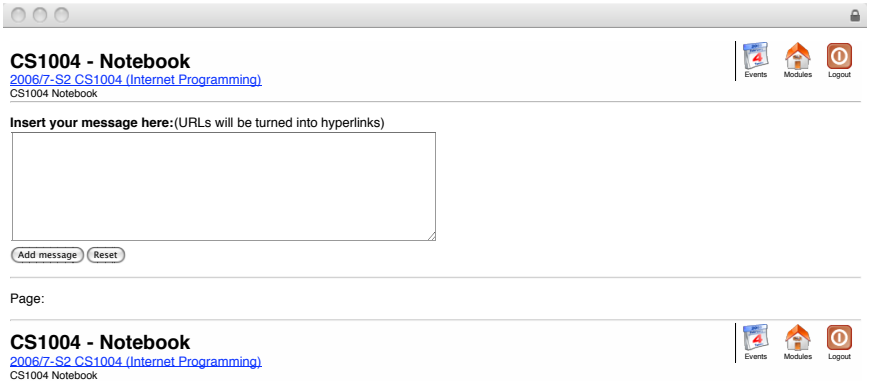


Figure 7.12: Screenshot of Notebook Tool

Data format compatibility was provided for the possibility of importing existing discussion threads into MMS. The tool user interface was retained as closely as possible to the original interface in TAGS, and provided a familiar tool for staff seeking the same functionality in new modules.

7.7.3 Forum Tool

There are obvious limitations to the notebook tool, most notably that it only supports a single thread of conversation for each tool instance. Lack of control over permissions restricts it to fairly simplistic use cases. It is unsuitable, for example, for staff to use it as a source of announcements. Discussions cannot be isolated to each group individually, meaning it cannot be used for discussion-based group work. The use of a flat file on disk means that the software must manage concurrency issues itself, and this particularly impacts performance under load.

The forum tool was the evolution of the notebook tool, adding support for discussion threads. Messages were stored in the database, with existing notebook tools were migrated to this new tool, using a convert-on-load approach, with their messages placed under a single thread. The tool could be configured to be read-only for students, and to restrict visibility of threads to groups or individual students.

7.7.4 Quiz Tool

As part of the migration from WebCT, a quiz tool was integrated into MMS as an option for replacing that functionality from WebCT. Quizzes were used in a very large number of modules at St Andrews, and are a popular feature of VLEs in general is that of delivering quizzes to students. This widespread use of quizzes can be seen in the variety of case studies involving quizzes in VLEs[78][97][115].

Initial development was driven by a request from the Department of Film Studies for a quiz tool in MMS, with the intent of discontinuing WebCT use for their courses. It was hoped that this feature might help encourage other schools to move to MMS for teaching functionality as well.

Unfortunately, the tool never achieved many of its goals. The tool was designed from requirements collected from Film Studies, without examining WebCT's existing implementation; this was done to ensure both that there was no suggestion of copying WebCT, and to provide a "blank slate" for the design, without any preconceptions.

The result however was a tool that did exactly what had been specified, rather than what was required. The requirements gathering process had been overly naive, as a result of overconfidence in the users' understanding of what they needed. The users had overlooked specification details that they felt were inherent to any quiz tool, and therefore had not considered in the design.

The core limitation was that it only supported automatically marked multiple choice questions, with no support for written answer, essay, calculated or other question types. Multimedia support for questions and answers was overly simplistic, making it difficult to integrate images, audio or video (the latter being particularly relevant for Film Studies) into questions. Lastly, tools for examining student answers and overriding them lacked many of the more powerful options of their WebCT counterpart.

The tool was used for a number of years in a handful of modules, but has never had significant take-up, and this is unlikely to change without a significant further investment in developer time. Further, as MMS at its core is an administration-focused platform, it is unclear whether adding such functionality is inline with the software's design, and may risk introducing unnecessary complexity.

7.8 Conclusion

MMS provides a wide variety of user-facing tools which address a substantial range of administrative processes, from management of assessment and attendance data, to reporting of data to central Registry. Underlying these tools are powerful configuration options to allow end-users to define and verify MMS' implementation of academic policies on matters such as grade calculation and transformation, marking scales, etc.

These administration tools are complimented by learning and teaching tools which enable users to remain within a familiar application for many common use-cases of a VLE.

Tools also expose multiple routes to achieving the same goals, for example the school-level tools for bulk administration of modules, module-level tools, student enrolments, etc. This ensures that users are equipped to approach problems in whichever workflow best fits their needs.

Chapter 8

Case Study: Academic Alerts

The next three chapters are case studies looking at specific examples of use of MMS. Academic alerts and postgraduate research student management involve administrative tasks which MMS was introduced to streamline and simplify, while the LAVA virtual archaeology project looks at use of MMS in teaching and research.

8.1 Context

During the course of a student's degree, there may be issues in relation to attendance, coursework submission, quality of work achieved, etc. Prior to the introduction of academic alerts at St Andrews, such issues were dealt with through a policy referred to as "Permission to Proceed" (commonly referred to as PtP or P2P). This policy was that disciplinary issues were dealt with by removing a student from modules they were taking. Students would generally receive a single warning that they were likely to fail, before being removed from the module or course if performance continued to be unacceptable.

This process had a number of drawbacks. Most significantly, it missed opportunities to intervene earlier in a student's academic career, when remediation would have been simpler. There was no standard process for notifying a student of issues that were not at the time likely to cause them to fail their degree, instead notification was solely that they were of imminent risk of failing. Appeals were also frequent, and resulted in significant paperwork both in managing these appeals, and having to manage consequential changes in case of

successful appeals (for example changing lists of students eligible to take exams)[136].

8.2 Introduction of Academic Alerts

The academic alert concept was designed in outline by VP: Teaching and Learning and Deans in 2010, to provide a standardised process for notifying students of academic related issues. These issues could ranging from minor issues with quality of work, through to immediate removal from a module on safety or ethical grounds. Unlike the previous procedure academic alerts were not included as part of a student's permanent record (transcript), in order to give academic staff the freedom to issue academic alerts without concerns of unduly negatively impacting a student's academic career. Academic alerts also did not (by themselves) prevent students from attending exams, in order to reduce bureaucracy related to managing attendance at exams.

8.3 Use of MMS

As a result of allowing and encouraging academics to raise concerns earlier and more frequently, it should be clear that academic alerts would inevitably result in a substantial increase in administration, due to significantly more alerts to be issued than there had been removals of students from modules. This is particularly true as a student can have multiple alerts for the same module, reflecting either multiple independent problems, or escalation of a problem over time.

These administration tasks would consist both of issuing academic alerts and auditing issued alerts (for example to allow staff to confirm that an alert has been raised). Naturally, any process which increases workload is likely to face adoption resistance, and accordingly it is desirable to minimise administrative impact.

MMS was used to refine and implement the academic alert process, automating issuing of alerts, re-issuing where needed and tracking issued alerts for reference. In providing tools for managing academic alerts, it also inherently enforces much of the process details (for example, limiting the types of alerts issued to approved options), and provides data for use in business intelligence analysis.

8.4 Process

Academic alerts are intended to be raised by a module coordinator or other senior academic involved with organising a module, on the basis that responsibility for administration lies with them. Given however that many of the academic alert causes are easily assessed, there is scope for delegation of much of this work to less senior staff. Examples of cases where this is appropriate can include a student missing a number of tutorials or lectures, failing to submit coursework in good time, being absent from a required component (such as safety training for Chemistry modules), etc.

The system is potentially further complicated by differences in the organisational structures of modules in different academic schools. It should be apparent that the logistics of administrating an English first year undergraduate module with 300 students is quite different to a taught postgraduate module in Computer Science with 20 students. While both may have a single module coordinator, in the case of a module with more enrolled students it is likely that more responsibility must be delegated to academics and administrators.

8.5 Issuing Alerts

There are three key stages to the lifecycle of an academic alert; problem identification, academic alert issuance, and auditing of alerts issued. MMS' existing functionality for attendance monitoring and coursework tracking was already suitable for assisting with problem identification.

A new tool was developed for issuing academic alerts, and embedded into the module administration pages. This provided a number of functions:

- Raising a new academic alert about a student.
- Issuing an academic alert to a student.
- Reviewing all issued academic alerts within a module.
- Re-sending academic alerts in case of lost e-mail.

Here “raising” an alert is the process of creating the alert, and can be performed by most academic staff. Issuing of alerts is generally restricted to module coordinator and senior staff in each school, and confirms the alert prior to it being sent to the student. This separation exists such that, depending on school policy, administrative staff could manage raising of academic alerts, with academics required only to verify the alerts before they are issued. The interface for raising a new academic alert is shown in figure 8.1.

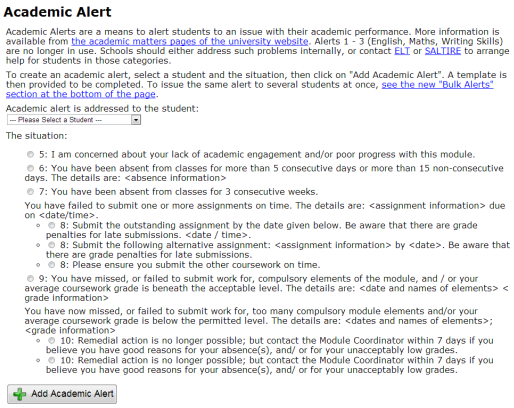


Figure 8.1: Screenshot of Raising an Academic Alert

To minimise work required while raising alerts, alert templates are provided. The design of the academic alert process limits the number of distinct types of alert to a well-defined set, which simplifies the design of these templates. Staff choose a class of academic alert to be raised, and a template from within that class, and then are only required to edit minor details such as due date of coursework, contact details for staff, etc. The core categories used are listed below:

- “I am concerned about your lack of academic engagement and/or poor progress with this module.”
- “You have been absent from classes for more than 5 consecutive days or more than 15 non-consecutive days.”
- “You have been absent from classes for 3 consecutive weeks.”
- “You have failed to submit one or more assignments on time.”
- “You have missed, or failed to submit work for, compulsory elements of the module, and / or your average coursework grade is beneath the acceptable level.”

- "You have now missed, or failed to submit work for, too many compulsory module elements and/or your average coursework grade is below the permitted level."

These categories have been expanded both to add subject-specific versions of the default text (for example ethics-related issues for Medicine), and to cover further issues (such as academic misconduct). Senior management within the university can edit these categories as needed.

In some cases, academic alerts need to be raised in bulk, such as where a major coursework deadline has been missed by a number of students (this can be more common on modules with high populations). Here the cause, action and risk statements are all identical, with the only difference being the student details. MMS provides a bulk issuance function for these cases, which requires staff select a list of students, complete details of the academic alert template, and alerts are mass-issued accordingly. The interface for raising academic alerts in a batch is shown in figure 8.2.

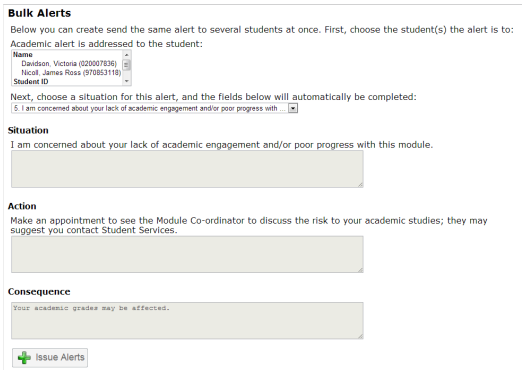


Figure 8.2: Screenshot of Bulk Academic Alert Raising

8.6 Presentation

Once entered, there is a need to view academic alerts during the life of a module. This is important for verifying lists of academic alerts sent, for ensuring students have been given sufficient warning and support in case of difficulties, for reference in case of appeals, etc.

This information is therefore integrated into a number of views through tools in MMS, including the coursework, exam mark, attendance and final grade tools, enabling at a glance reference to whether alerts have been raised for a student.

8.7 Institution-Level Management

The last component of academic alert support was to allow senior management to manage and analyse the process. Academic alert types are defined generally by the Vice Principal for Learning & Teaching, with input from the Deans. Rather than having these changes written out simply to be entered into MMS by someone else, an interface is provided to directly allow these members of staff to directly alter the categories.

To provide feedback on the processes effectiveness and usage, MMS can export statistics on numbers of alerts issued, broken down by type, date and academic school.

8.8 Conclusion

Almost 3,200 Academic Alerts were issued in the academic year 2012/3. Without MMS the Academic Alert system could not have been embedded in academic processes, meaning adoption would have been infeasible due to increased administrative workload.

A version of Academic Alerts for postgraduate research students was later designed and approved, by popular demand from academic staff, but has not yet been implemented due to resourcing constraints in postgraduate research student management support.

Chapter 9

Case Study: Postgraduate Research Student Administration

9.1 Context

The number of postgraduate research (PGR) students at St Andrews has shown consistent increases in recent years (this is detailed in table 9.1 at the end of this chapter), and therefore an increase in administration effort required to manage them.

Common administrative tasks for managing a postgraduate research student's academic lifecycle include submission of annual reports, thesis submission, leave of absence, etc. These tasks place significant demand on academic staff, especially those staff with oversight responsibility (for example Pro Deans). This is compounded by variations in processes and lack of clear audit trails to manage issues.

The Pro Dean for Postgraduate Science requested that tools were introduced to MMS to simplify these tasks. Looking at areas where tools could be readily introduced into MMS' framework, the following pieces of functionality were developed:

- Modify the existing coursework tool to better the use-case of support school-internal progression reports by students and supervisors.
- Add a new tool for managing student and supervisor annual progress reports to the

Pro Dean.

- Add a new tool for supporting workflow for nominations of external examiners for thesis examination, and to prepare forms for the examination.

9.2 Annual Internal Reports

Monitoring of postgraduate research students' progress at St Andrews includes a pair of annual reports from the student and their supervisor to senior academics within the school. Each student's report reflects on the year to date, and the corresponding supervisor's report addresses the student's progress through their degree and any concerns arising. These reports are also read by the student's second supervisor and reviewer.

MMS' coursework tool already handled submission of files by students, however support for progress-orientated marking and staff-only reports needed to be added. Extending the coursework tool to add this functionality instead of developing a whole new tool retained the coursework tool's extensive feedback support, auditing processes, and almost a decade of testing. Progress-orientated marking here refers to the marking scale used for postgraduate research student reports, which uses a traffic-light scale (red/amber/green) to reflect confidence in a student's progress.

To fulfil this use-cases staff-only assignments were added as a new assignment type (adding to the existing options of file upload and paper hand-in), and a traffic-light marking scheme added to the abstract mark API (see section 5.11.3).

9.3 Annual Report to Pro Dean

In addition to the internal reports, supervisors and students are expected to independently submit reports to Registry, for review by the relevant Pro Dean. This is an opportunity for students to raise any concerns about their degree, supervisor, or even their academic school if relevant. Historically students engage poorly with the process, and it is common for them to mis-interpret the intent of the process and mistaking it as an extension of in-school progress monitoring. This can lead to a missed opportunity to resolve any issues arising

with their supervision.

There were several problems with the existing approach:

- Pro Deans are responsible for around 400 students each. In theory this should mean 800 reports are sent to each Pro Dean, although lack of student engagement means this number can be significantly lower.
- There was no way to automatically check that schools had in fact returned reports for all of their students. Student records had to be manually consulted, which was a time consuming and error-prone process.
- There was no way of matching up the independent student reports with the school reports such that the relevant Pro Dean could see the reports along side each other. Registry had to hire extra help into process the piles of paper and manually reconcile the pair of reports for each student. This was not only costly but highly inefficient with delays of up to 9 months between reports being sent to Registry and being presented to the Pro Deans. Given that the reports are time sensitive, any issues raised in them may have become critical before the Pro Deans read the report.

The reports themselves are structured as Word document forms, with a number of elements on both the student and supervisor forms that are nearly identical. Moving to a pair of online forms was therefore a clear way to improve the process. Students and supervisors were given separate forms to fill in, and the submitted forms correlated and presented to the relevant Pro Dean. Resubmission of forms (for example to correct errors, or in exceptional circumstances) was handled by automatically pairing the most recent two forms.

Once completed forms are presented side-by-side to the Pro Dean, allowing them to compare related answers on each form to identify problem areas. Electronic forms also made it simple to verify whether all students (and in some cases supervisors) had returned the forms, and contact them where appropriate.

9.4 Nomination of External Examiners

Nomination of external examiners for research students is a task that is both needed for all students, and requires time from senior management (one of the Pro Deans). As such, it is a clear opportunity to increase efficiency. The task also requires entry of data that can at least in part be validated automatically (for example completion of all relevant fields, format of e-mail address, etc.), and has a multipart workflow (sign-off by a number of different staff in different roles).

A single page web form was developed to capture the required data, as well as providing workflow control for approval. Once the form is completed, it can be passed along for acceptance/rejection by management within the school, before being returned to the Pro Dean for final approval. The age of MMS' design has meant this process was unexpectedly complicated to develop, however. While the input validation framework is well suited, lack of an object relational mapping layer (Hibernate, SQLAlchemy, etc.) meant that data being persisted to the database required extensive amounts of SQL to be manually written.

9.5 Examination Process Forms

The next logical step after addressing nomination of external examiners was to look at the later stages of the examination process. Initial intent was to reimplement the full workflow in software, although uncertainty about feasibility limited this. While under normal circumstances a student would proceed linearly through a number of steps from arranging for submission through to graduation, in cases such as requiring major corrections to a thesis, or difficulties nominating an external examiner, the process required some steps to be repeated. Additionally each step required to be audited individually. Unfortunately with limited time available it was felt that it was more practical to focus on supporting tools rather than a redesign.

From discussion with Registry and academic staff two of the most significant causes of time taken are entering details into the existing forms, and correcting errors where such details are mis-entered. To streamline this process, functionality was added to MMS based on the same technology used for coursework coversheet generation (see section 7.4.1), to merge data into the forms as Word documents from central records.

9.6 Conclusion

In the first year of the new process for submitting progress reports to the Pro Dean (academic year 2010/1) there is a substantial increase in student report return rate (see table 9.1). Note that this table shows full time PGR students against each year they are registered for, and reports against the year the report is submitted in. There are more school progress reports than students in 2008/9 and 2009/0 due to reports being submitted late from previous years, and counted against the year submitted in.

An increase in return rate by students is clearly visible, with around three times the number of reports returned in 2010/1 compared to 2009/0. Reports are now of course instantly available to the Pro Dean once submitted, drastically improving timeliness of the report and accordingly any response to issues raised.

Academic Year	Student Reports	Supervisor Reports	Full Time PGR Students
2008/9	80	657	625
2009/0	155	775	648
2010/1	475	690	754

Table 9.1: Return Rates for Progress Reports to Pro-Dean

Informal feedback on the external examiner nomination process suggests that it has significantly reduced scope for errors, especially submission of incomplete forms, although the process to set up and navigate the tool is more complicated than would be ideal. Lastly, there has been very limited feedback on the examination process forms, but initial impressions would appear that without more significant work the changes to date do not provide sufficient motivation to overcome the inertia of existing workflows.

Chapter 10

Case Study: LAVA

10.1 Context

The Laconia Acropolis Virtual Archaeology project (LAVA) was a research project by Kris Getchell[31][32], attempting to provide a virtual fieldwork environment for students studying archaeology. Places on actual fieldwork assignments for archaeology students are limited, meaning that many students are unable to partake of a key tool for gaining hands-on experience.LAVA provided game-like tools for allowing students to experiment with planning and executing of a dig, and exploring dig sites.

The LAVA project consisted of multiple different tools, many of which required functionality MMS was well placed to support. MMS acted as a control plane for LAVA, providing it with authentication and authorisation (in particular management of lists of students on the module), discussion forums, group work support, and progress tracking. Three new tools were developed by researchers working on the LAVA project and integrated into MMS:

1. Archaeological dig management simulation, covering the planning of the dig, hiring of staff and purchase of equipment, and management of resources as the dig progresses. Access to a 3D version of the dig site was also managed through this tool, although delivered through software independent of MMS.

2. Group work tool for allowing students to work collaboratively on a shared document, in this case a dig proposal and funding application.

3. Map exploration tools, for allowing students to navigate through maps of archaeological digs. This was used to allow them to explore the site they were proposing a dig for, and see examples of actual artifacts found on the site. See section 10.6.

All of these tools are discussed in detail later in this chapter.

Some adaptions were also made to the MMS core to add a structured workflow around the tools students would use, so that they had to look at the site before they could start writing their dig proposal, and complete the proposal before they could start their actual dig.

10.2 Use of MMS

The tools made extensive use of MMS' student management functions, pulling group information from MMS to determine which students were working together, as well as to check students had access to the tools, authenticate user access, handle authorisation for administrators, etc. These tools also used the rendering framework provided by MMS for data tables (Flex). This reduced development effort, while providing students with an interface style they were already familiar with.

10.3 New Developers

LAVA is notable as it was the first time researchers outwith the main MMS team were involved with development within the MMS framework, and provided an opportunity for those researchers to provide feedback on the MMS API. The two researchers working directly within MMS are referred to here as R1 and R2 (working on group work and site tour tools respectively).

Feedback from the two researchers was limited, in part because at the time I did not consider the value of formally collecting feedback. However, some feedback was provided, and additionally for quality control and security purposes both researchers' code was reviewed in detail by myself, and that provided some insights.

While neither developer appeared to have significant issues with how to use the API, it

became apparent that there had been a failure to effectively communicate best practises. R2's work in particular reflected the overly-simplistic explanation of the Flex rendering tool as being intended for use in displaying tables, which had her using the Flex for all tables, not just those containing data (contrast with tables used for layout). This could be mitigated in future by providing both worked examples and demonstrating use cases when briefing new developers.

There were lesser issues related to data suitability, in that the design of the dig simulator presume users would always want to see either the student or staff interface of the tool, and for students to belong to a group. Unfortunately staff (especially developers) may need to act as students for the purposes of testing code. Students were also not necessarily in a group (they may not yet have been allocated a group) and there was no clear way of handling that in the design.

MMS native tools were designed with these constraints in mind, and their design provided access to student interfaces for staff (modified to be read-only to avoid causing possible problems through staff submitting work for modules they are not students on). No other tools critically required students to be part of a group for them to work. These could be worked-around through staff users switching to student accounts for testing, and the tool would present users with an access denied message if they were students without a group, but this was fiddly and overly complicated. Were there to be work to add further tools of a similar nature later, there would need to be careful consideration of how to better address these issues.

10.4 Student Workflow Modelling

One other issue arose about fitting the LAVA tool types into the MMS framework, which was the need to provide a rigid workflow to the tools. Most tasks in MMS contain material constrained by dates by which they must be completed (coursework to be submitted, lectures to be attended) and not by any form of prerequisite. The archaeology tools on the other hand were distinct but belonged to a structure; trying to write a proposal before touring the site would not make much sense, and until funding had been allocated by teaching staff it would be impossible to start excavating the site.

Accordingly each tool in MMS given two new attributes, a boolean indicating whether the tool was considered 'completed' and a reference to a tool which must be completed before this tool was accessible (where applicable). The former was determined by the tool as needed, the latter stored as a database field. When rendering a user's homepage tools were loaded for display as normal, and where a prerequisite existed the preceding tool was checked for completion. This allowed chains of tasks to be assembled, although the initial implementation was inefficient as number of database queries scales linearly with number of tools the user has access to.

This feature appeared to over-complicate presentation of tools; students generally could follow a numbered list of activities, while staff who did not require the functionality found the option of a prerequisite tool added unnecessary complexity. The ability for misconfiguration to cause significant problems for students, which were unclear in how they were caused, meant that in the end the function was removed.

If this functionality was needed at another point, options for improving clarity should be considered. Suggestions include hiding the option under an "Advanced Options" tab, and providing clearer indications to students as to why a tool was unavailable (as well as diagnostic information for staff to check student progress). It is notable that Moodle has similar functionality[83]; in our experiences similar confusion does arise, although in the context of an application intended primarily for teaching (as opposed to reducing overheads including support) this appears to be less problematic.

The other approach we considered for managing workflow was to have a "super-tool" that wrapped around the individual tools for each stage. Wrapped tools would be permanently hidden from the default view of a module, and students would have to access the super-tool to get to the individual workflow elements. This may have presented a clearer user interface, but would have required extensive architectural changes for a relatively small number of users, and would have customisation for each workflow.

10.5 Dig Site Management

Development of the site management tool was initially by R2, continued by R1 after R2's departure, and later completed by myself. The tool presents an excavation as a series of

dig ”stages”, with a set of requirements that students must fulfil before they can move onto the next stage. Example requirements include ”Remove the topsoil” or ”Hire a surveyor”. To complete these requirements, students, working together in groups, can choose staff/equipment to hire/buy for use in the excavation. Staff include diggers to perform the bulk of the work, specialists to analyse artifacts found, and support staff such as cooks and computer technicians to provide support to the diggers/specialists. Each group is assigned a budget for the dig, which is set by teaching staff based on work submitted through the collaborative form.

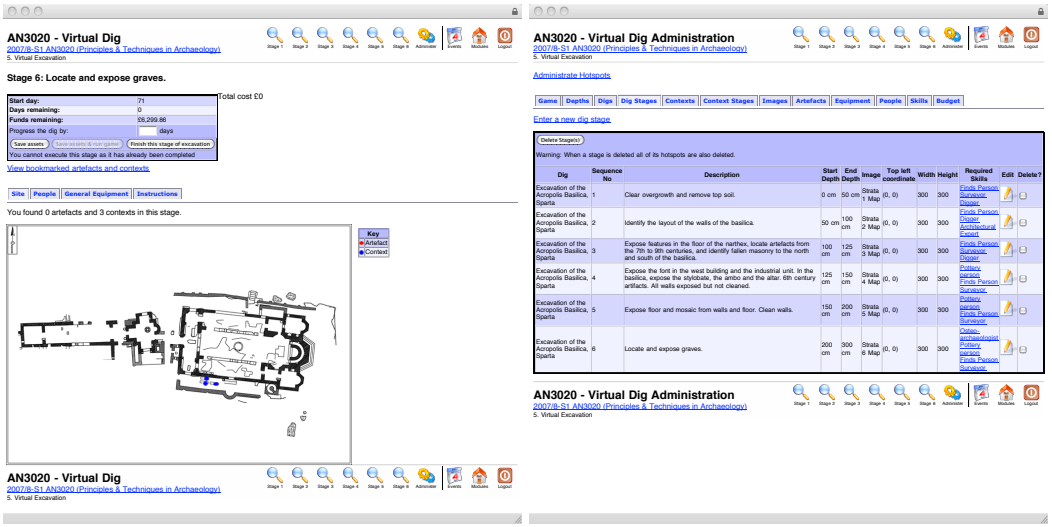


Figure 10.1: Screenshots of Archaeological Dig Management

As the dig progresses artifacts are dug up, and may be also identified, depending on staff/equipment assigned. As with a real dig, students must make a note of where each artifact is found, and may also add their own notes on the artifacts. Currently only one set of game logic used to determine if artifacts are found is implemented, but the resource is designed to allow alternative game engines to be plugged in.

10.6 Virtual Site Tour

Ahead of starting an archaeological dig it is conventional to tour the location to establish sites of interest. The virtual tour resource allows students to take a tour of a dig site, presented in the form of set of interlinked maps. Maps are provided from a high level overview of the site, and can be zoomed down to small points of interest. In contrast to

other web based maps such as Google Maps, users are limited to a specific area, however the displayed maps are prepared with archaeological information included.

The tour was implemented originally by R1 using Flash to produce the pages (Flash was specified in the original requirements), and later rewritten by myself using HTML and CSS to produce an equivalent interface with a simpler implementation. The virtual site tour tool is shown in figure 10.2.

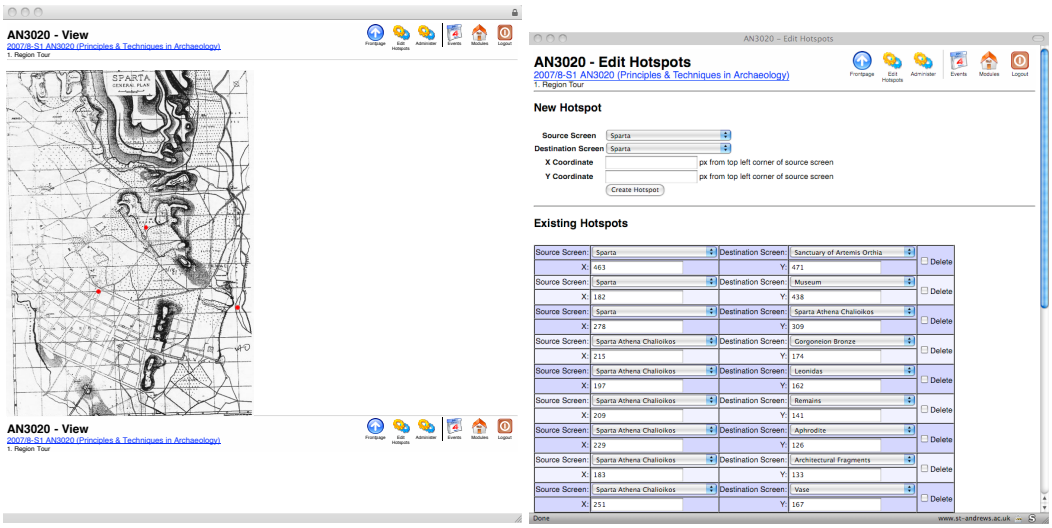


Figure 10.2: Screenshots of Virtual Tour

In LAVA, a tour of the excavation site is provided to students before they submit their funding proposal, allowing them to get a basic feel for the location, and an idea of the buildings and artifacts they might find during the excavation.

10.7 Collaborative Form

A funding application is generally required to secure funds for a dig. The collaborative form is intended to facilitate this step, letting students work together to answer a set of preset questions. The collaborative form was designed and implemented by R1. The tool has been designed with support for questions in the form of freeform text, although future work could include options such as multiple choice (particularly useful for automated assessment of submitted answers). The collaborative form is shown in figure 10.3.

It is noteworthy as the first tool which provides revision control for submitted work, and

the only tool where students can view previous versions of work (this is normally restricted to staff). This functionality is intended to support group work in an asynchronous tool, allowing students to track changes made not just by themselves, but by others in their group, and revert changes if necessary.

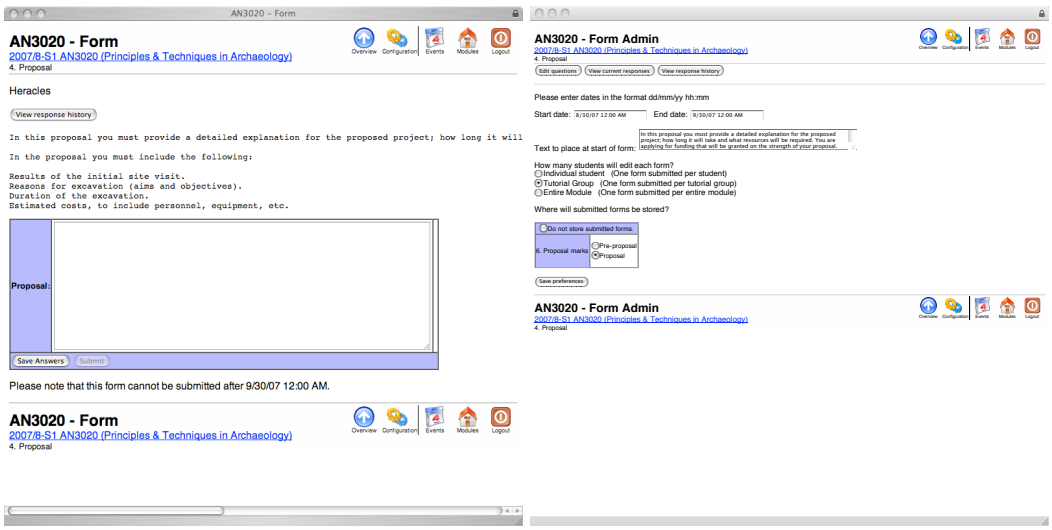


Figure 10.3: Screenshots of Collaborative Form

Answers are only sent to staff for marking once all students in a group have accepted the most recent version. This ensures that no single student can disrupt the group by submitting their answers without taking into account the other student's opinions. Once answers are completed, the form tool can automatically prepare a PDF document of the answers and submit it to an associated coursework tool for marking. This integration was performed via a generic API exposed by the coursework tool, which could also be used to deliver work to other tools if the need arose.

10.8 Use of MMS Data in External Learning Tools

The LAVA researchers were experimenting with 3D virtual worlds for allowing students to explore dig sites and/or re-creations of historical sites[32]. One attempt to allow students to access to these worlds involved launching the 3D environment from MMS. This was to be done both to ensure that the launch process could be simplified as much as possible from the student's perspective, and to allow the progress of a student through the exercise in MMS to affect the 3D world.

The Jake[1]) 3D environment was used in this case. Jake is a re-implementation of the Quake 2 engine in Java, and was chosen both due to familiarity with the Java language, and the ready availability of tools to manage Quake 2 datasets.

The launcher was added to the LAVA dig simulation tool within MMS, which uses Java Web Start[2] to launch the Jake application and pass details of the students' progress through to Jake.

This progress data determined the "map" (virtual world) that was loaded by Jake, such that students were presented with an appropriate stage of the dig. These stages covered a variety of scenarios from clearing topsoil through to having exposed much of the structure.

The implementation at this point was only developed to the point of providing a one-way data flow from MMS to Jake. If further developer resource was available making the data exchange bi-directional would be an obvious improvement, for example to allow students to modify the dig in a 3D environment and have those changes reflected in the dig simulation tool.

This work also lead naturally into other work in virtual worlds, such as Second Life. MMS' functionality for generating Second Life identities (avatars) was exploited by an HCI module in 2009[102] to support association of students' real identities (and student records) with their Second Life identities for assessment purposes.

[1] `http://bytonic.de/html/jake2.html` - accessed 15th June 2013

[2] `http://www.oracle.com/technetwork/java/javase/javawebstart/index.html` - accessed 17th February 2015

Chapter 11

Adoption and Evaluation

This thesis seeks to answer whether academic administrative overheads can be reduced through the use of web applications. To assess MMS' success in reducing costs, feedback is collected and analysed from staff users of MMS. These users are best placed to evaluate changes in time taken for tasks they undertake, given depth of experience with the processes before, during and after the switch to MMS.

MMS' adoption across the university has been primarily voluntary rather than led by senior management, and as such acceptance by staff is also a key requirement for success. In this regard usage statistics from a number of sources are evaluated, to show adoption over time and as such the staff-led demand for MMS.

Additionally, MMS' architecture and tools will be evaluated based on my own experiences and that other researchers who have worked with the software, in order to assess its suitability as a framework for hosting applications.

11.1 End-User Surveys

In order to assess acceptance and usefulness of MMS a number of surveys were performed at points in the application's lifecycle. Composition of the surveys is based on the System Usability Scale[17] (SUS). The System Usability Scale is a set of 10 questions, 5 with a positive phrasing, and 5 with a negative phrasing, which respondees indicate their

agreement/disagreement with. The normal process for calculating an SUS score from these results is:

"To calculate the SUS score, first sum the score contributions from each item. Each item's score contribution will range from 0 to 4. For items 1, 3, 5, 7,and 9 the score contribution is the scale position minus 1. For items 2, 4, 6, 8 and 10, the contribution is 5 minus the scale position. Multiply the sum of the scores by 2.5 to obtain the overall value of SUS."

- Taken from Brooke[17]

Higher values are better, however this score is not intended to be interpreted as a single data point. Convention is to compare scores over time in order to illustrate change, rather than using it as an absolute measure of usability.

Survey audience in 2006/7 was limited to Computer Science, however later surveys were open to all students and staff at the university, who were notified of the surveys as part of weekly memos.

11.1.1 Questionnaire Summer 2006/7

At the end of the academic year 2006/7 Computer Science students and staff were invited to complete a web-based survey of MMS. Limitations in the survey software meant that a scale of 1-10 (strongly disagree to strongly agree respectively) was used rather than the conventional 1-5 range for SUS. Unfortunately as a result there was no mid-point "Neither agree nor disagree" option, which would normally be present in an SUS survey. The questions and results from each question are presented in table 11.1.

Given the range differences in this survey compared to SUS, the calculation of the final score required modification in order to produce a score out of 100. To do this, score contributions from each question were summed, and multiplied by 1.11 (100 divided by 90). Score contributions from odd-numbered (positively phrased) questions used their value minus one, and from even-numbered (negatively phrased) questions was ten minus their value. Responses containing unanswered questions were ignored (accounting for 15 out of 81 responses).

Final scores ranged from 12.22 to 100 inclusive, with a mean of 75.45; this value is of

Question	Mean
I think that I would like to use this system frequently	7.03
I found the system unnecessarily complex	3.06
I thought the system was easy to use	7.89
I think that I would need the support of a technical person to be able to use this system	2.1
I found the various functions in this system were well integrated	6.56
I thought there was too much inconsistency in this system	3.62
I would imagine that most people would learn to use this system very quickly	7.45
I found the system very cumbersome to use	3.03
I felt very confident using the system	8.12
I needed to learn a lot of things before I could get going with this system	2.29

Table 11.1: Survey 2007 Results

limited value in isolation, and will be compared with the results of later surveys.

11.1.2 Questionnaire Winter 2009/0

An extended survey was performed across all staff and students after the first semester of the academic year 2009/0, to gather feedback on system effectiveness and assist with resource planning in terms of development effort. The timing of this survey was chosen as it was at the end of the first semester after which MMS was rolled out across the whole institution.

The core text of this survey is reproduced in appendix A for reference. The SUS questions were included (under “Usability”), and corrected to offer 5 options (Strongly Disagree through to Strongly Agree). Additional a number of other questions were included, looking at user base demographics, how the system is used, its effectiveness, and priorities for future work. Breakdown of user roles is shown in figure 11.1.

The results of the survey elements on changes to staff time expended are shown in figure 11.2, reflecting a majority of positive or neutral responses.

The SUS score on this occasion was lower than the 2006/7 score, with an average of 69 taken from the 84 responses. Unfortunately in expanding the audience for this survey, it's impossible to be certain how much this can be attributed to a worsening of the user experience vs move away from a technical audience. This figure accordingly primarily provides context for the 2012/3 survey, which was delivered to a similar audience and assessed on the same scale.

The 27 written responses provided in the "Other" field for future priorities cannot be reproduced as-is due to the anonymous nature of the survey, but are summarised below. 4 were actually complaints about the survey interface intentionally not allowing them to pick multiple items at the same priority (of potential relevance to any future attempts to use such a survey, that it needs to be clearer and/or simply that respondees will hate the limitation).

Many, if not most made reference to navigation, layout, and other user interface issues, and in a few cases illustrated confusion over system behaviour (for example a comment that students cannot replace uploaded work – they can, but only before the submission deadline or by approval of staff). Attempts were made to rectify this through clearer user interfaces in later years, however it is clear this is an area of weakness. The research team has struggled to produce and maintain up to date user guides, both due to limits on time available and development pace.

The remaining issues raised reflected limitations in the implementation, and have since been addressed. Specific examples include support for courses running outside of the standard academic year, and for different lateness penalties.

11.1.3 Questionnaire Summer 2012/3

A final survey was performed in summer 2013. Building on experiences from the previous surveys, this was split into three individual surveys, each targeting different groups of users. Staff and students within the university each were given similar but separate surveys, and external examiners were also surveyed in order to provide an alternative perspective by users best placed to compare with other systems and processes at other institutions.

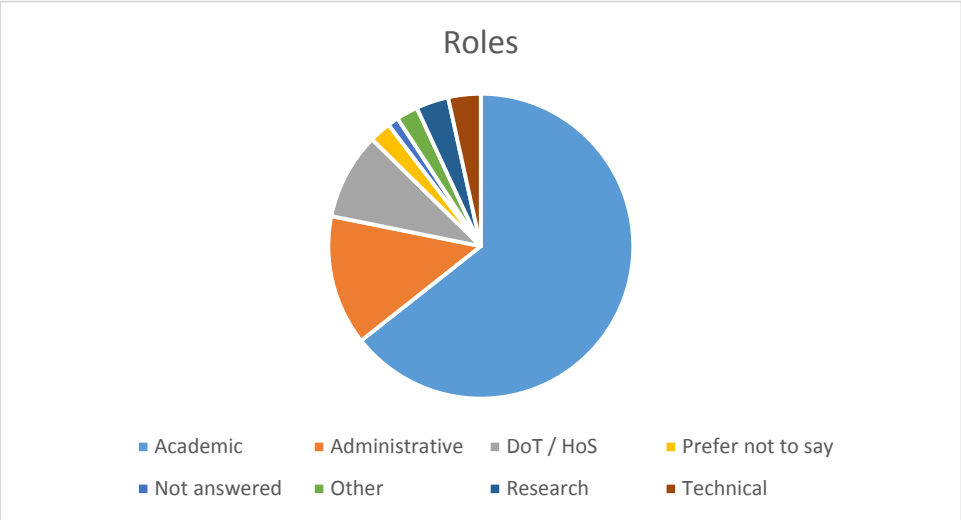


Figure 11.1: Staff Roles (2010)

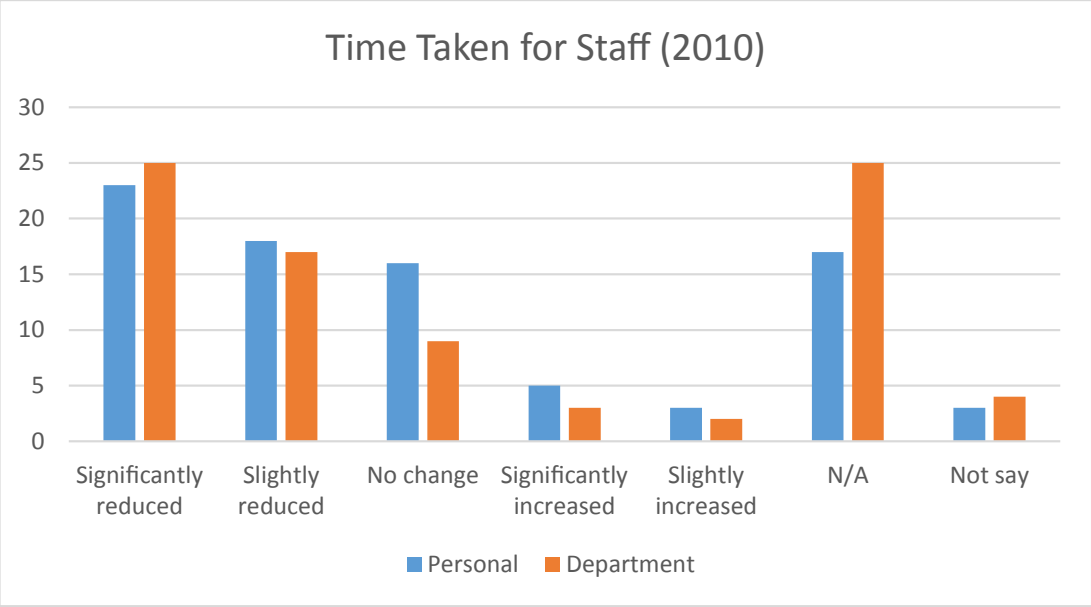


Figure 11.2: Staff Time Taken (2010)

In the case of users who are both students and staff, the instructions specified that they should complete both surveys. The text of this survey is reproduced in appendix B.

SUS

All three surveys included a section with SUS questions, in order to provide a comparison with the previous surveys. As with the 2010/1 survey this was performed on a Strongly Disagree to Strongly Agree scale, and the score calculated in line with the normal SUS process as described by Brooke[17] (and reproduced at the start of this section).

Results for all categories, with previous scores for comparison, are shown in table 11.2. Scores for both staff and students improve on the 2010/1 score, with students showing the most significant improvement. However the score for external examiners is dramatically lower, and their feedback on MMS is discussed in detail later in this section.

	2007	2010	2013 (Staff)	2013 (Student)	2013 (External)
Score	75.45	69	70	77	44
Responses	66	84	25	45	4

Table 11.2: Survey 2013 SUS Results

Staff

Staff are the main customers for MMS, although they are not the majority users of MMS (accounting for about 15% measured both by number of users, and number of tool-accesses over an academic year). There are two key groups of staff that are expected to benefit from MMS; academics and administrators. For both groups MMS seeks to reduce time taken on a number of administrative tasks (this actually means that the more effective it is, the less academics should be using the system), and this was the main focus of the staff survey.

Figure 11.3 shows a breakdown of staff responses to the questions “If relevant, how do you feel MMS has changed the time taken for you to prepare and submit final module results?” and “In generally, how do you feel MMS has changed the time taken for your department/school to prepare and submit final module results?”[1]. Here you can see the

[1]Question reproduced as-is, including incorrect use of “generally”

majority of responses were “Significantly reduced”, with no cases where staff reported an increase in perceived time taken.

Student

Although students make up the bulk of the active users of MMS, their usage of the system primarily benefits others (staff), and therefore acceptance of the system is of primary interest, rather than benefit to them. Students were asked “Looking ahead to the next academic year, would you like to use MMS on more/less modules you are taking?” and “Still looking ahead to the next academic year, would you like to see more or less tasks (such as coursework submission, return of exam marks, delivery of course content) make use of MMS?”. Figure 11.4 shows students’ responses to these questions, split into “Wider” and “Tasks” categories respectively.

As with the staff survey above, responses have been positive, with the majority of respondees indicating a desire for the same or more use of MMS. However, there is a clear majority happy with the current level of usage of MMS. This is not entirely unexpected given that students are not the key audience, however it could indicate scope for further improvement in student-facing functionality.

External Examiners

A further survey was sent out to external examiners. Unfortunately the number of responses was extremely low (four), and therefore cannot be considered a statistically significant sample. However, the written comments provide insight into the difficulties experienced and the low scores given as a result. Of the two examiners who provided written feedback, both showed common themes of preferring the use of paper for the process of external examination, and of frustration with time taken to navigate the system.

This suggests that while the design captures requirements of staff and students well, lack of engagement with external examiners during the early design stages has meant the user interface does not meet their needs.

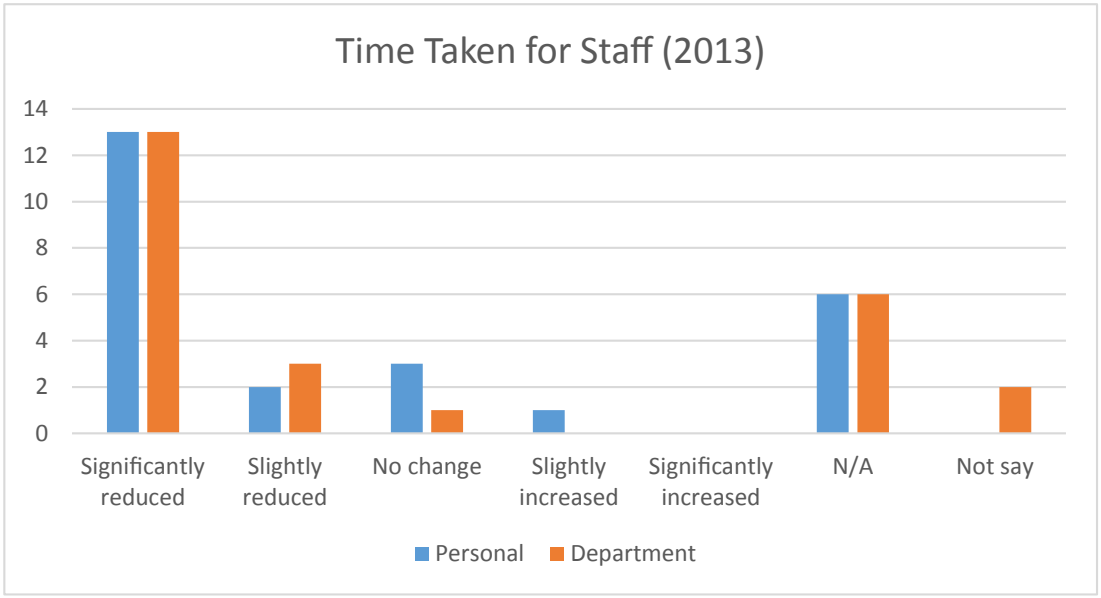


Figure 11.3: Staff Time Taken (2013)

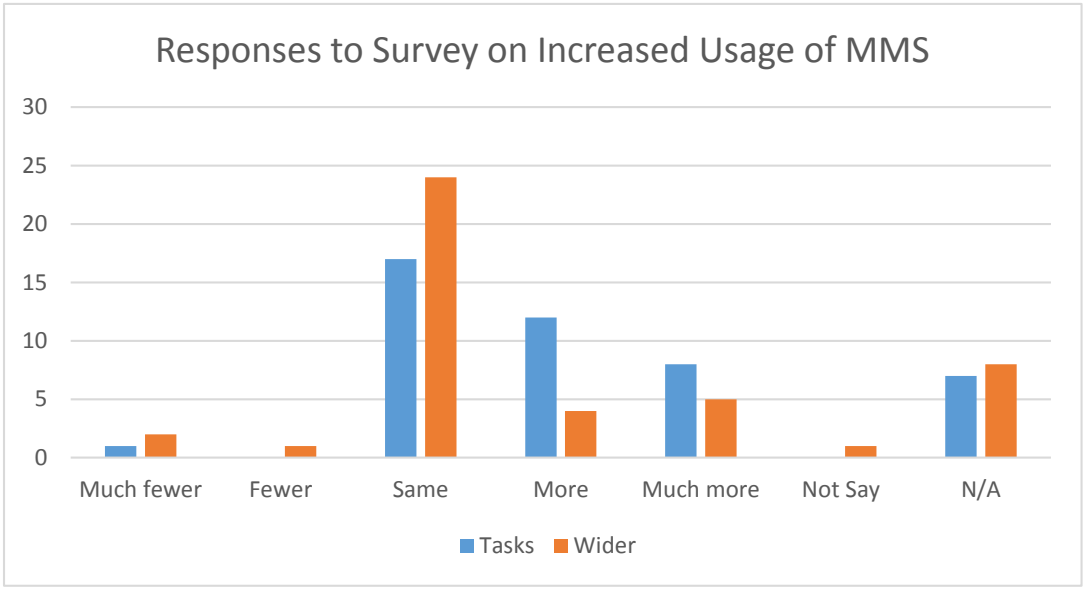


Figure 11.4: Student Change Desirability

11.1.4 Summary

Feedback from users of MMS with regards to usability has been generally positive, and very positive in terms of time saved by the system.

The very limited number of responses to surveys by external examiners is disappointing, particularly given as a group they have extensive experience of other institutions' systems and processes. The survey likely suffered from timing, having been sent out after examination meetings had been completed, and re-attempting the survey with distribution prior to examination meetings may yield better results.

11.2 End-User Interviews

Following on from the surveys performed in previous years, in 2014 staff from a number of academic schools were interviewed on their usage of MMS with the intent of gathering feedback on the system. All interviews were held via email at interviewees' request (in person, by phone or Skype options were offered and universally rejected). The ethics committee required a list of questions to be asked prior to interviews taking place, and as such questions were static.

Interviewee selection was designed to find those most involved in using MMS, in contrast to the earlier surveys which were broader in scope. It was anticipated that given the time taken for an interview, the number of responses would naturally be lower, and heavier users would be more invested in responding. To this end the list of staff to invite to interview was taken from the list of primary contacts for MMS. This list has arisen based on those members of staff who actively communicate with the researchers, typically either senior teaching or administrative staff. Typically one member of staff "represents" each academic school within this list, although depending on school structure in some individual departments may be individually represented.

Out of the contacted members of staff, 11 replied that they were willing to take part, and of those 9 have completed the interview questions. Interviewees have been assigned a single letter between A-K inclusive to identify their responses here. Letters have been assigned randomly, and gaps in the sequence indicate interviewees who have not responded.

11.2.1 Questions

The questions were chosen to gather background information on the interviewee, to better understand the context of their answers, to understand how MMS affects their work, and to gather feedback on what could be improved. The questions are reproduced in full below for reference:

- Would you describe your role in using MMS as primarily academic, administrative or technical?
- Please outline the main tasks you use MMS as part of, and how you MMS in this context.
- Overall, how would you say MMS affects the time taken for these tasks?
- How does MMS, if relevant, assist with these tasks?
- How does MMS, if relevant, hinder these tasks?
- In your opinion what are the best and worst aspects of MMS?
- What would you want to change about MMS?
- Any other comments on MMS?

11.2.2 Roles and Main Tasks

The intent of the first two questions is to establish how the interviewee primarily uses MMS, to provide context for their answers to later questions:

- Would you describe your role in using MMS as primarily academic, administrative or technical?
- Please outline the main tasks you use MMS as part of, and how you MMS in this context.

Full responses are reproduced in Appendix C. Answers to the first question were:

A. Academic

B. *No response*

C. Administrative with very occasional technical work

D. As School Administrator, I am involved in working with all aspects of MMS in an all-round administrative capacity (Unit/Module Administrator role for all School modules).

E. Administrative

F. Primarily academic - as a lecturer on a range of modules, across ugrad and PGT levels.

G. *No response*

H. Administrative

I. Technical with administrative

J. Academic

K. Administrative

Around two thirds of interviewees indicated their roles are primarily administrative, with most of the rest being academic. This contrasts with the responses from the broader survey in 2010 which saw a primarily academic response.

In response to the question about tasks, most interviewees listed administrative tasks, with notable exceptions of interviewees F and J. Administrative tasks here are differentiated from academic tasks in that they do not provide teaching value directly, but support teaching, for example collection of assessed work or monitoring attendance vs delivery of taught materials. Although interviewees were generally extremely positive about how MMS affects time taken, it is notable that several listed setting up modules, and adding staff to modules as tasks, and these are overheads introduced in the use of MMS or similar systems. The impact of these tasks is mitigated by the ability to reuse this data in other systems such as VLEs or the student survey tool, however ensuring that such overheads are minimised is important.

Responses to the question on usage predictably map well to the tools listed in chapter 7 with common themes being configuration of MMS (this appears often, and suggests motivation for further improvements in streamlining the processes), coursework, attendance monitoring, exam mark entry and reporting of final grades. Accessing disability information also appears prominently on many lists. One unanticipated area of high usage is in sending emails to sets of students based on module or tutorial group, reflecting positively on re-use of enrolment data present in the system.

11.2.3 Effectiveness of MMS

The next three questions address how effective MMS is at reducing time taken. The questions are intentionally similar, and phrased in a number of ways to try reducing positive/negative bias in answers. The effectiveness questions were:

- Overall, how would you say MMS affects the time taken for these tasks?
- How does MMS, if relevant, assist with these tasks?
- How does MMS, if relevant, hinder these tasks?

Answers were overall extremely positive about the impact of MMS, reflecting that MMS simplifies most tasks and reduces time taken. The negative responses are more insightful about future work, and discussion below will focus on those answers. The noticeable difference between the wider surveys in previous years and this interview emphasises that MMS has been tailored to the needs of administrative tasks, and suggests potential need to consider a broader range of users when planning future work. Interviewee J does specifically cite setup time as an issue.

Common themes in the responses refer to eliminating need for spreadsheets, reducing risks and scope for error, and bringing data into a single system.

Responses in regards to MMS as a hindrance question cited difficulty in navigating larger modules, as well as handling complex configurations. For example interviewee A's response highlights navigation issues:

"Sometimes if MMS module sites become cluttered by poor management then finding important information could involve opening many windows and the information might be within a folder, within another folder within another folder etc which takes time to get to."

This has similarities to feedback from external examiners in the 2013 survey, where time taken to navigate within MMS was a significant source of frustration.

Interviewees C and H refer to issues with complex module configurations, with C describing a case where custom scripting is required to support the configuration, and H describes an unusual assessment diet where optional units of work are given different values, which MMS does not currently support. Although a marking script could be written to support such a scenario, the logic for deciding how many pieces of work are required is limited to n of m scenarios for fixed values of n and m.

11.2.4 Feedback on MMS

The final set of questions are intended to gather feedback on MMS. In many cases interviewees provided significant feedback as part of earlier questions, although repeating the same question with different phrasing did yield some further detail. The questions posed were:

- In your opinion what are the best and worst aspects of MMS?
- What would you want to change about MMS?
- Any other comments on MMS?

A range of common themes became apparent from responses, with significant positivity about the quality and scope of data available via MMS, it's ability to automate common tasks, and effectiveness in replacing previous spreadsheet-based solutions. The user interface was both praised and criticised, suggesting issues with consistency of quality of the interface. Specific issues were emphasised in relation to navigating large data sets (content, students, groups, etc.)

A number of interviewees specifically requested improvements to the content tool, citing desire for a more aesthetically pleasing interface such as those found in WebCT or Moodle.

This and feedback on the lack of up to date guides and slow pace of bug fixes emphasised the impact of the relatively limited resources of a research team in maintaining large pieces of software.

Some items of feedback reflect issues which are significantly more difficult to address, where they are the result of decisions outwith the researchers' control. This is particularly true of data related feedback, for example details of students taking resits is not available from central data sources after the student has completed the resit, and keeping this data available would require finding a solution to rationalising differences in the data sets in cases where data quality errors cause students to have incorrectly listed as resitting. These areas of feedback are well suited to further examination in future work.

11.2.5 Summary

Overall feedback was extremely positive, with the issues raised primarily the result of lack of resources (both development and technical documentation). This supports the conclusion that web applications can be used to reduce administrative overheads, and that MMS has succeeded in doing so.

However there is clearly room for further improvement in navigation and handling of bespoke configurations. In the context of navigation improvements it is suggested that any work both looks at directly improving navigation (for example better search functions and interconnectivity between data items), and at reducing need to navigate by introducing new views which address workflows which are currently poorly served (such as external examination).

Handling of custom configurations could perhaps by improved by hiding implementation details from the end-users, and/or by looking at whether the rollover system could better handle such cases. It may also be that some cases currently handled through scripting should be re-considered as built-in functionality.

11.2.6 Addressing Challenges

It is desirable to show that MMS overcomes the challenges detailed in chapter 3, however this is difficult to do as it amounts to showing the lack of these challenges in using MMS. It is possible to illustrate this through the changing content in feedback on the system between the earlier surveys and the interviews described above. In the 2010/1 survey the written responses highlighted issues in process modelling, while in 2014 navigational issues have become the major concern, showing progress in the system design. This is especially true for academic process implementation (i.e. lateness policies), and mode of attendance handling (and the interactions in those differences), which are explicitly mentioned in the 2010/1 feedback as issues.

11.3 Web Server Access Logs

This section will examine activity in MMS based on the web server access logs as an indication of adoption patterns for MMS.

Web server access statistics are extremely simple to record, in that most web servers provide logging functionality by default, and are easily understood. However they are of limited usefulness for showing adoption due to the effect of content caching in browsers and/or proxies. This caching can mean requests are not sent to the remote web server by the user's browser, resulting in an indeterminate (although typically small) error margin[36]. Some methods have been proposed for overcoming this limitation through changes to the HTTP protocol[106], however these have not been widely adopted, and are not used here.

Additionally, in the case of MMS there is a significant gap in the recorded data, due to lack of proper archival policies in place during the replacement of server hardware. Accordingly, these access logs based results are primarily presented as simple breakdowns of usage, providing an introduction to details that will be analysed in depth from other sources.

11.3.1 Methodology

Access logs for MMS are recorded by the Apache Tomcat server, and these logs are produced in either the "combined" or "common" log formats. Logs have been parsed and inserted into a MySQL database for ease of data manipulation. Resulting values have been exported from MySQL in CSV format, imported to Excel and charts generated by Excel.

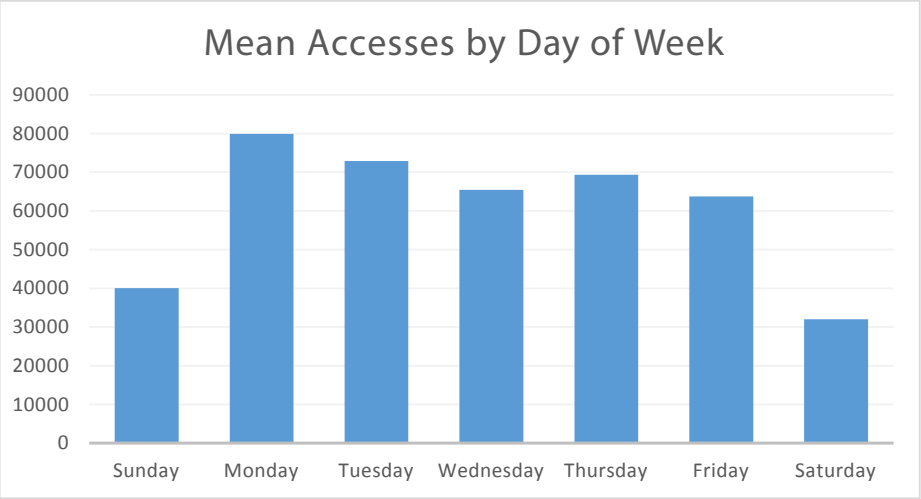


Figure 11.5: Access By Day of Week

There is an inherent variation of activity in institutional systems across time. This is most clearly evident in accesses across days of the week, as can be seen in figure 11.5, which shows requests received by MMS by day of week. The sample period is across a number of academic years, although the intent is to show relative values rather than absolute values. Here you can clearly see that the bulk of activity is across Monday to Friday, with weekends being predictably quieter.

Further evidence of this variation can be seen by comparing semesters within each academic year; figures 11.6 show activity in semesters one and two of a number of different academic years. The structure of semester two differs from semester one, in particular that there is an extended break during the semester, causing a substantial dip in activity. While there is a week's break in semester one (referred to as "Reading Week"), this does not appear to have the same impact.

This variation reflects the variation in structure of each academic year and semester; in particular semesters are not necessarily the same length (number of weeks) across different academic years, nor is teaching necessarily delivered at the same pace or in the same order.

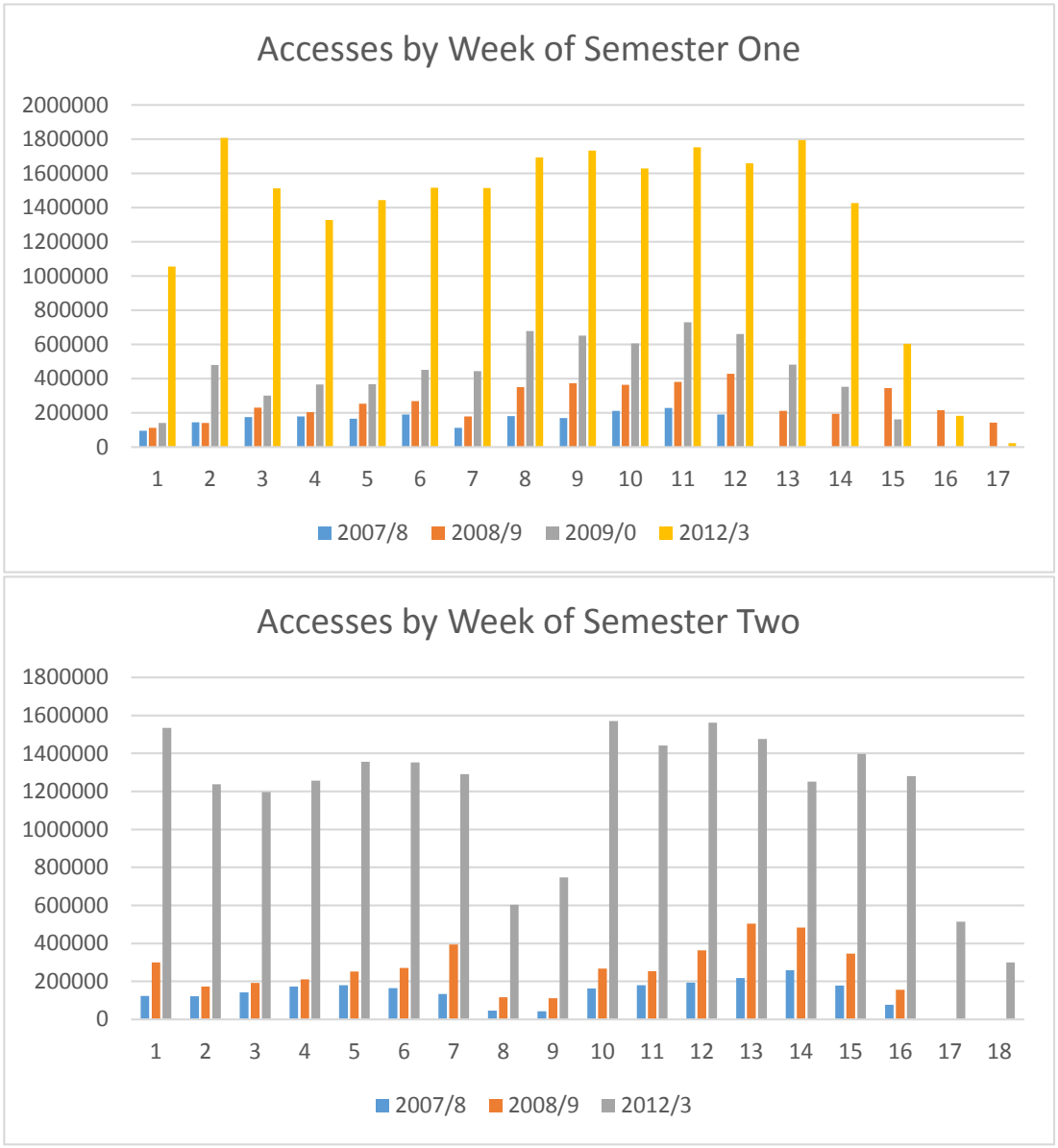


Figure 11.6: Access By Week of Semesters 1 & 2

11.3.2 Access in First Week of Semester One

It is desirable to use data from points in the academic year which are as consistent as possible in activities undertaken, in order to minimise variation due to external factors, as highlighted above. Accordingly the data presented focuses on the first week of each semester.

In the first week of semester one, there is a consistent pattern of activity within the institution, wherein students are signing up for modules and groups, looking at the structure of the academic year, and staff are generally focusing on last-minute changes to setup where required. First week of second semester is similar, although typically fewer changes in terms of student enrolments at this point.

HTTP requests received by MMS in first week of semesters across a number of academic year is shown in figure 11.7. As can be clearly seen, accesses in the academic year 2012/3 are significantly higher than 2009/0, and accesses have shown consistent increases over time.

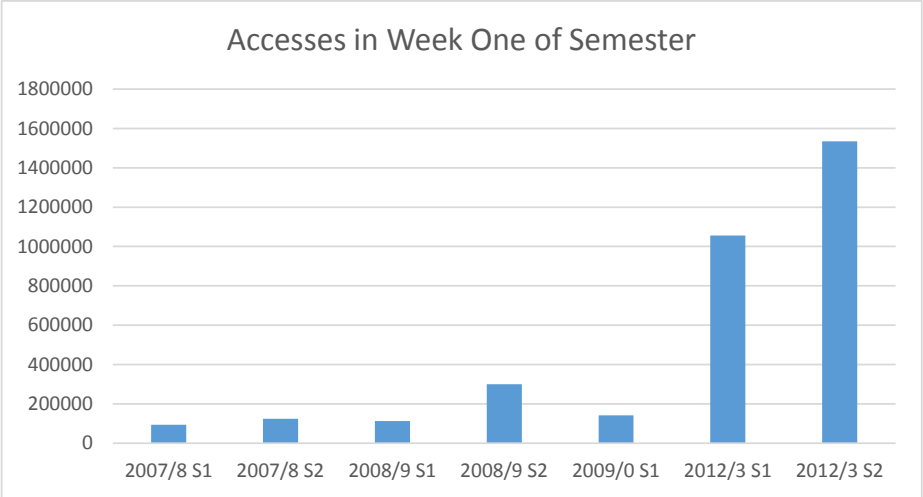


Figure 11.7: Activity in Week One of Semester One

11.3.3 Status Codes

The access logs also record the HTTP response code returned to the client, which can indicate the outcome of a request. This data for the years 2007/8-2009/0 is presented in

table 11.3; where no semester is specified, measurements are during the time outside of a normal semester. Specifics of each status code can be found in the IANA HTTP Status Code Registry[2].

	2007/8			2008/9			2009/0	
		S1	S2		S1	S2		S1
200	683,701	1,727,515	2,255,164	802,820	3,550,674	3,310,515	183,279	4,740,759
206	14	206	7	9	39	90	27	273
300							1	106
301	2,594		15,514	18,297	85,532	99,943	10,441	728,579
302	8,561	134,330	204,405	51,521	282,897	401,177	749	5,306
304	24,281	178,856	303,247	120,490	473,745	583,125	39,003	1,399,342
400			177	1	33			
401				3		218	79	9
403	60	2,880	1,372	746	3,296	1,391	82	179
404	42	427	769	576	501	2,063	16,281	47
405	2	391	937	45	49	67		
416								1
422	5	241	169	47	178	157	23	2
500	71	267	508	403	1,585	755	183	237
501				2	2	2	4	23
503	8	463	240	104	38	21	1	905

Table 11.3: HTTP Response Statuses by Academic Year

Of particular note are the records for the status codes #200 (OK), #403 (Forbidden), #404 (Not Found) and #500 (Internal Server Error). #200 codes indicate requests which have succeeded, #403 and #404 codes indicate problems with the user's request and #500 indicate an error within the server. With further data, it might be possible to infer effectiveness in URL design in MMS to enable users to find content or tools they are looking for. However the extreme spike in #404 responses in 2009/0 compared to previous academic years does not have any clear meaning without further data.

The #500 status codes however can be used to indicate the number of errors sent back in response to requests; to see this spike with the increase in usage is expected, however the

[2] `https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml`

later decrease towards the end of the recording period, in contrast to the ongoing increases in usage, supports the idea that MMS' code was stabilising during this period.

The lack of long term reliable data as well as details of what the errors were limits analysis options at this time, however this would be one opportunity for future work.

11.4 Usage

This section will examine usage statistics recorded within MMS, to build on the summary data extracted from the web server access logs. Again here a spike in usage in 2009/0 is driven by a change in university policy, but otherwise changes are driven by staff within each academic school.

MMS maintains its own logs of user activity, which provide a more narrowly focused view of usage of MMS. The activity logs are primarily designed to track users for the purpose of auditing changes; for example, when students have claimed they uploaded work in time, and MMS "ate" their coursework, the logs can be used to show the students activity around the time in question. This can help confirm whether students have logged in or not, identify cases where they have uploaded work to the wrong place, etc.

11.4.1 Logins

User logins are one of the easiest metrics to understand. Unlike measurements of requests, they are guaranteed to be activity by a "real" user (rather than an automated system such as search engine's spiders). There are variations based on how users use MMS – someone who logs in daily to check for changes will inherently show up as more active than someone who logs in once a week. None the less, this data is clearer in its meaning than using simply raw request logs.

Table 11.4 and figure 11.8 show this data, using the first week of each semester as the measurement period. Logins are measured as total number of logins, rather than as logins by unique users. Data prior to semester two of 2009/0 has not been recorded to the same level of accuracy, and therefore is not presented here.

	2009/0	2010/1	2011/2	2012/3
S1	-	37057	42938	42035
S2	14536	36962	39489	39157

Table 11.4: Logins in Week One, by Semester and Academic Year

As can be seen in the data, there is a dramatic increase in activity from 2009/0 to 2010/1, and a smaller increase from 2010/1 to 2011/2, however surprisingly there is a slight dip from 2011/2 to 2012/3. This is particularly unexpected given that student attendance figures at St Andrews have increased steadily over this time period, which would naturally drive a higher usage.

The significant dip from first semester to second is a potential indication of cause – while second semester would generally entail less setup and change during its first week, the dip (around 8%) suggests that individual users are also simply logging in less often. This could indicate a decrease in need to login as users become more familiar and as such comfortable with the system.

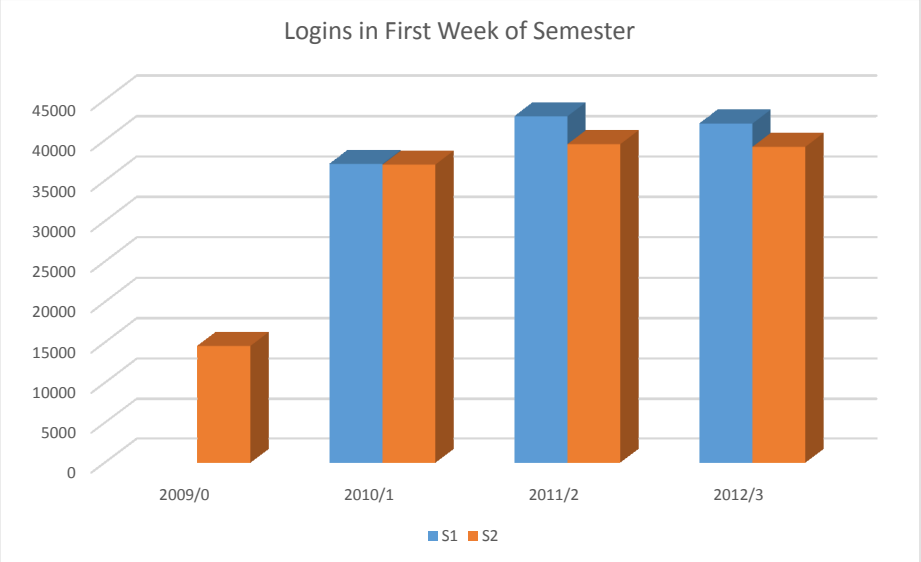


Figure 11.8: Logins in Week One, by Semester and Academic Year

11.4.2 Modules

Table 11.5 lists a breakdown of modules by academic school and year, showing the pattern of adoption across the university. Note that this table only reflects academic schools for brevity, and where non-schools have created modules there may be a discrepancy between this table and total modules in the academic year. Generally a steady increase is shown until adoption plateaus, with the exceptions of English Language Teaching (ELT) and Modern Languages. Both of these schools have provided positive feedback, and it's unclear why the number of modules has dropped in 2010/1 for Modern Languages and 2012/2 for ELT.

As stated elsewhere the jump in 2009/0 reflects a change in university policy, however adoption in other years is a voluntary change on the part of the schools.

	2005/6	2006/7	2007/8	2008/9	2009/0	2010/1	2011/2	2012/3
Art History					43	48	59	54
Biology				10	100	107	117	157
Chemistry					61	60	70	70
Classics			1		87	90	79	77
Computer Science	46	53	54	64	60	58	56	64
Divinity			1	6	46	63	56	72
Economics and Finance		23	19	67	63	63	65	63
English Language Teaching			1		22	25	27	15
English					75	73	74	82
Geography and Geosciences				43	82	85	83	106
History				160	154	163	150	166
International Relations					85	93	93	95
Management					71	82	70	70
Mathematics and Statistics		40	38	44	75	68	76	64
Medicine					12	16	16	21
Modern Languages				55	278	185	195	212
Philosophical & Anthro Studies		35	58	83	94	101	110	122
Physics and Astronomy					71	76	78	88
Psychology		22	28	35	56	67	67	85

Table 11.5: Modules by School and Academic Year

11.4.3 Tools

Table 11.6 shows number of modules using tools in MMS across a number of years. The core tool types (coursework, exam and final grade) have been highlighted at the top of the table. Total number of modules in MMS are shown at the bottom of the table. This data commences from 2005/6 as that was the point at which support for multiple academic years was introduced, and prior to that point a new installation was created for each academic year.

	2005/6	2006/7	2007/8	2008/9	2009/0	2010/1	2011/2	2012/3
Coursework	41	103	156	430	1120	1227	1266	1342
Exam	27	84	94	280	876	908	926	913
Final Grade	40	100	121	411	1414	1403	1420	1496
Content	4	24	27	44	149	838	897	1032
Dissertation Titles						36	37	22
Enrolment	12	10	13	18	12	12	10	12
Group Signup				24	91	230	293	339
Lecture Attendance	10	7	4	8	16	5	2	3
Moodle Link				1	15	496	493	584
Notebook	5	1		2	3	21	61	50
PGR Admin						26	30	27
Quiz					5	10	9	9
Tutorial Attendance	15	17	36	155	442	482	509	556
URL	10	10	23	33	71	95	215	196
WebCT Import				3	10			
Modules	46	174	201	576	1541	1535	1556	1695

Table 11.6: Modules Containing Tools, by Academic Year

Usage through the recorded period has increased by a factor of 30-40, reflecting the high rate of adoption of MMS. The spike in usage in 2009/0 was driven primarily by a policy change in reporting of module results, however outside of this adoption has been driven at the individual school or module level. A subset of this data is also shown in figure 11.9, please note that while the main spike in adoption is in 2009/0, that there is a significant increase in both 2008/9 and 2007/8.

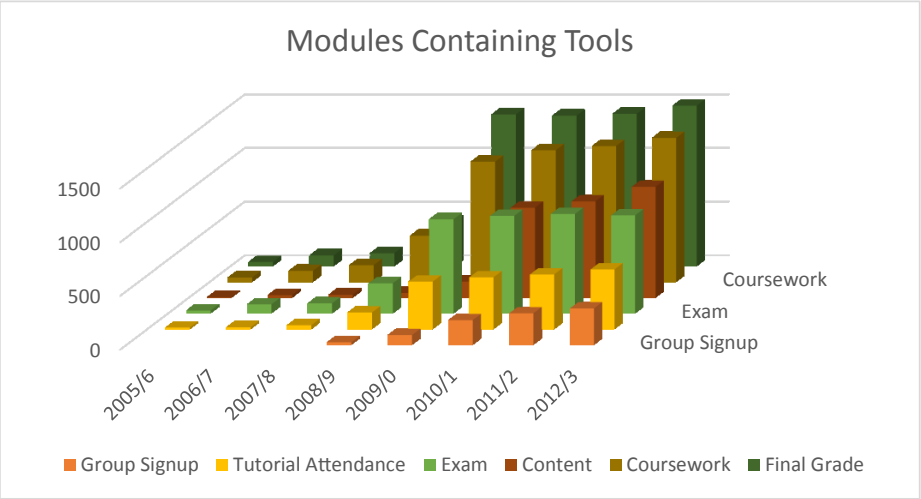


Figure 11.9: Modules Containing Core Tool Types

11.4.4 Coursework Assignments

Table 11.7 shows coursework assignments in MMS for the academic years 2005/6–2012/3. A subset of this data, showing first and second semester only, is shown in figure 11.10. The two assignment types "Paper" and "Upload" reflect work where no file is to be uploaded to MMS (it is only used to store the coursework marks), and where a file is uploaded, respectively. Note that this "Paper" does not inherently mean electronic submission hasn't been performed outwith MMS, and "Upload" does not necessarily imply a paper copy may not be required in addition, no data is available on how widespread such hybrid scenarios are.

Semester	Style	2005/6	2006/7	2007/8	2008/9	2009/0	2010/1	2011/2	2012/3
S1	Paper				151	905	613	542	485
S1	Upload	149	206	274	285	441	996	1118	1318
S2	Paper			48	262	910	694	608	486
S2	Upload	130	202	203	286	535	988	1114	1367
Y1	Paper			3	194	247	157	139	108
Y1	Upload	50	95	40	50	166	560	678	720
Total		329	503	568	1228	3204	4008	4199	4484

Table 11.7: Assignments by Academic Year

There is a clear increase in assignments in MMS over time, with a predictable leap in 2009/0 in line with the general adoption of MMS across the institution. Further, while the total number of assignments climbs over time, the number of assignments where submis-

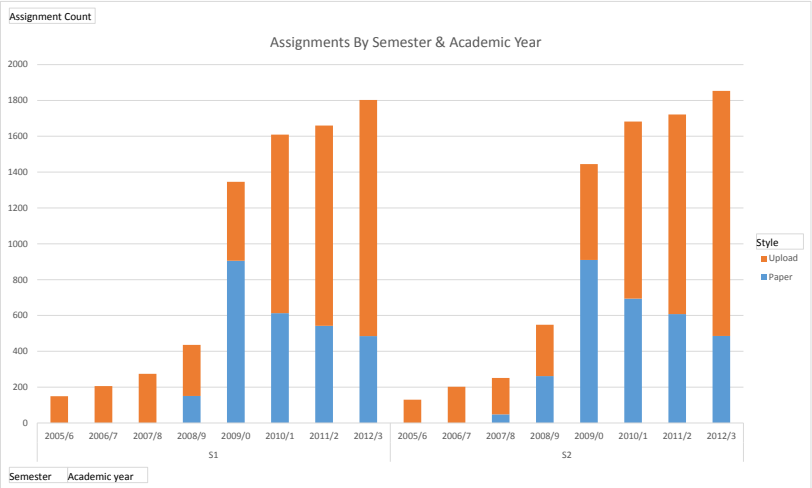


Figure 11.10: Assignments by Academic Year

sion is on paper and MMS only stores the mark peaked in 2009/0 and has steadily decreased over time since then. This reflects the increased usage of MMS for electronic submission of coursework.

A breakdown of the number of assignments set by academic year is shown in table 11.8. Number of assignments submitted in total and to Turnitin are shown in table 11.9. Again, a consistent increase can be seen in usage, reflecting staff demand for the system.

Academic Year	Assignments Set	Turnitin Enabled	%
2007/8	517	0	0%
2008/9	621	30	5%
2009/0	1142	503	44%
2010/1	2562	1762	69%
2011/2	2922	2376	81%
2012/3	3446	2665	77%

Table 11.8: Total Assignments and Assignments using Turnitin, by Academic Year

Academic Year	Assignments Submitted	Submitted to Turnitin	%
2007/8	22629	0	0%
2008/9	41322	1087	3%
2009/0	99509	12552	13%
2010/1	104680	34043	33%
2011/2	107239	46878	44%
2012/3	110335	49802	45%

Table 11.9: Total Assignments and Assignments using Turnitin, by Academic Year

11.5 Researcher/Developer Feedback

MMS' framework is designed to support developers, and therefore experiences of those who have developed within the framework is an important element of the evaluation. To date this is limited to the MMS researchers themselves, and the researchers working on the LAVA project, where MMS was used to provide authentication, authorisation, rendering support and workflow, as well as data interchange with other elements (for example submission of work for assessment from the collaborative form tool to the coursework tool).

11.5.1 MMS Team

Feedback within the MMS Team has unsurprisingly been positive, given the framework was designed and has evolved around their needs. None the less, as will be discussed in more detail in chapter 12 (Future Work), a number of popular frameworks have been developed since the MMS project was started, while the age of the MMS framework is beginning to show.

Issues are beginning to arise both in user expectations about interactions with software, and in training time for new developers being brought onto the project, whereas finding developers with experience in Ruby on Rails, Spring or node.js is relatively straight-forward.

Compared to the previous TAGS project the framework is substantially more advanced, robust, and reflective of the underlying structures (both data and organisational), and it is clear that MMS could never have succeeded without it. Accordingly future work in this area is highly encouraged.

11.5.2 LAVA

Feedback from the researchers on the LAVA project was informal and reflected a small subset of the overall LAVA project, however in general was positive. The MMS framework isolated them from the complexities of dealing with institutional data sources, enabling them to rapidly develop new functionality which utilised information on students and groups. The ability to integrate their tools into a software environment with which students were already familiar was a further benefit for simplifying adoption.

However there were some issues with lack of support for workflow handling, and of poor explanation of how to effectively use some of the APIs. MMS was not designed for tools to have prerequisites, for example for LAVA it was desirable that students could not access the dig tool before the funding application had been completed. Some guidance was provided to students through ordering of tools, however in isolating the tools from each other it proved challenging to provide a framework for handling such scenarios.

In relation to the API, intent of some components had been poorly explained. For example the researchers believed they had to use the table rendering API for all UI elements, leading to an extremely table-heavy design. Improved documentation and worked examples would be of assistance here.

11.6 Summary

Use of MMS has increased dramatically over the last decade, reflecting its widespread adoption within the University of St Andrews, and its incorporation as a key element of the university processes.

Overall this shows MMS' success in streamlining and minimising administrative tasks, and proves the theory of using web applications for this purpose. This clearly shows the value of tools which model administrative processes and a framework which closely models the institution. Given the challenges described in chapter 3, the supporting framework is instrumental to enabling these tools, and they would be infeasible to maintain and support, potentially even to develop, without it.

Chapter 12

Future Work

Implementation of MMS was started in 2002, and while substantial effort has been made to stay up to date with innovations in technology, naturally the focus for the limited development resources available has been on meeting core requirements. Essentially the code has aged[100], and it may be that future work could examine a significant rewrite/redesign in order to "catch up" with newer technologies. There are also a number of less drastic design changes and features which have been considered during the project but resources have not been available to address, and are proposed for further research.

This chapter is split into five sections:

- "Design" – high level design of MMS, and lessons of general applicability
- "Usability" – ways to improve the user experience
- "Organisational Architecture" – ways of improving the modelling of an institution
- "Software Architecture" – improvements of the core software architecture and technology of MMS
- "Features" – new features suggested for related work or future versions of MMS

The design section highlights lessons learnt from MMS which are of relevance if designing new related applications, as opposed to extending or replacing MMS itself. Usability

directly tackles issues raised in evaluation and how they could be addressed. The organisational and software architecture sections examine ideals for improvements to the MMS framework to simplify the design and improve functionality. Lastly, the features section will discuss potential new functionality for the user tools, and user-facing functionality in general.

12.1 Design

There are a number of lessons learnt from MMS which should be kept in mind for any future work. Significant value has been derived from the approach of closely modelling the institution, rather than using a more generic model and requiring constant effort to "translate" between the model and the real structures, and this should be emphasised when designing applications which focus on administrative processes. Re-use of data, from the institution, within the application, and from the application into other systems, has dramatically improved the usability of MMS and rewards for time spent managing its data.

Development of the application is substantially simplified by supporting scripting, allowing domain experts to implement policies without the overheads and communication risks entailed by central development resources being used for these tasks. A suitable test environment for these scripts is obviously required to ensure implementations are correct.

A defence in depth design both to security and error handling has reduced support overheads to a level where it is feasible (although arguably not desirable) for the researchers to support the application with over 9,000 active users.

12.2 Usability

One of the most recurring themes throughout the feedback from end users has been that the ability to navigate MMS, especially for moving around within a module, needs improvement. The existing design was based on an idea that the user logs in to MMS for a single activity, be to apply marks to work, or to download lecture notes, and does not sufficiently address the needs of users to navigate between tasks. Further, the user interface metaphors do not scale well to the quantity of material now present in MMS, and improved tools are

needed to managing searching MMS.

Some work has been done on mass editing tools, for example to entire marks for multiple students as part of a single assignment, and one option would be to look at extending these views. These could also form the basis of improvements for external examiners, who frequently require to compare data sets such as marks assigned for pieces of work.

12.3 Organisational Architecture

MMS' model of the institution is designed around a classic top-down organisational structure, with limited ability to reflect other cross-cutting groupings, or complex arrangements of modules. There are teaching elements arranged by subject, such as safety courses for Chemistry first years, which are not well served by the current structure. Postgraduate research students are typically managed by degree course (or within a set of similar degree courses), which MMS currently simulates using module-like entities. As experiences from TAGS illustrated, models should accurately reflect the underlying data rather than "re-imagine" it to fit a software design.

Some modules are taught together, either partially or wholly, and MMS also lacks the ability to merge modules in this manner. This stems from MMS' design seeking to accurately reflect organisational structure in the internal data architecture, however from an end-user point of view causes duplicate effort managing such modules separately.

A number of areas for improvement to the model are outlined below.

12.3.1 Modelling as Parallel Hierarchies

The original idea of modelling the institution's organisation, taught modules and subgroups as a homogeneous hierarchy was overly simplistic, in that it was neither an accurate or simple representation of the entities. However, it is believed that a suitable compromise can be found between this and the current relatively rigid model. Consider the entities as belonging to one of the following three classes:

Organisational Faculty, school, department

Teaching Module,“compound” modules, tutorial group

Degree Degree type (B.Sc, M.Phil, PhD, etc.), degree course, subject

Each of these classes is appropriate to different roles of user; management and administrative staff are more likely to be interested in viewing data across an organisational entity, while taught students are more likely to be interested in specific modules, and admissions staff and research students generally work across degree courses. These are not necessarily simple hierarchies, in that entities may have multiple parents (see cross-cutting concerns, below) however in most cases each entity has a single parent.

For all of these classes, roles assigned at a higher level apply to all entities directly below that level. There are inheritance points across hierarchies as well, for example modules belong to schools/departments (although it is unclear where “compound” modules fit in) as do subjects (as part of a degree course). Figure 12.1 shows a simple version of these hierarchies, containing a sample set of degrees, schools and taught modules; solid lines indicate relationships within a hierarchy, dashed lines relationships between hierarchies.

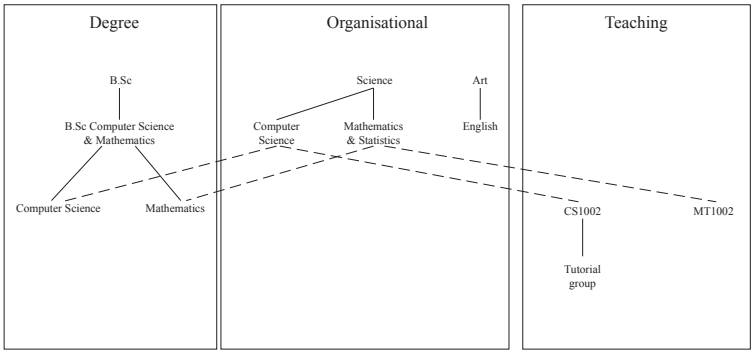


Figure 12.1: MMS Alternative Organisational Model

Tools (coursework submission, exam marking, etc.) and users (as students, staff or otherwise) would then be added to fairly arbitrary points on this hierarchy. These relationships would need annotations to indicate details such as roles, as well as handling modelling of cross-cutting concerns.

12.3.2 Cross-cutting Concerns

There are number of role-related details which cut across the normal hierarchy, such as level of module (1st, 2nd, 3rd, 4th, sub-honours, honours, postgraduate), distance/non-distance and full-time/part-time attendance. These are applicable both to modules within the hierarchy (potentially also degrees), and to user relationships with those entities.

For example, it is common at St Andrews for a school/department to have a sub-honours coordinator, who requires access to all taught modules at the sub-honours level. Their role is assigned at the school/department level (because that is the most appropriate point), but inherited access would then be filtered by criteria specified on the role. Another example would be staff dedicated to supporting part-time/evening students across the institution as a whole, who would be assigned a role at the institution level in the hierarchy, but with filtering specified to restrict applicable modules to those taught on a part-time/evening basis.

There is therefore value in having support for cohorts of students based on the students' membership in existing groups within MMS, ideally for tools to be provided to such cohorts as they are to students on modules.

12.3.3 Cross-institution Support

It must be noted that degrees may be taught with contribution by other institutions, following a number of possible models. Some courses may involve external staff working closely to teach and mark local students, or students travelling to external institutions for parts of their courses. This can be a cause of additional complexity, as data is frequently managed independently in both institutions' systems. The ability to model cross-institutional teaching would therefore be of benefit, perhaps by extending the suggested organisational hierarchy detailed above to handle multiple institutions at the top level.

This would also benefit from adoption of federated authentication, in order that staff from recognised external organisations can readily access the system without requiring to have an account manually created. This is particularly useful for external examiners, who otherwise would face having a large number of different accounts for different institutions they are examiners for.

12.4 Software Architecture

While MMS already has a powerful framework for supporting its core operation there are still a number of potential routes for future improvement. The implementation also dates back to 2002, and as such for many of the services provided (data abstraction layer, HTML rendering, input validation) have more recently developed alternatives which may be more appropriate for any new development or major reworks.

It may be that a complete redesign on a framework such as node.js or Ruby on Rails could further benefit development. node.js is of particularly interest given the use of server-side scripting in MMS, and could simplify extension of this functionality to other elements of MMS.

The following subsections suggest more minor software architecture extensions based on the existing design:

12.4.1 Object-Relational Mapping with Version Control

A significant proportion of MMS' current code is used to manage database interactions. MMS was developed initially without the use of an object-relational mapping framework, and instead uses hard coded or runtime-generated SQL to perform the vast majority of its database operations. Some effort has been made towards adopting Hibernate, however the widespread use of version controlled data within MMS significantly over-complicates this process.

The current database version control methodology is discussed in detail in the design chapter (section 4.8.1), however in summary two tables are used to store data with version information. The first table stores the underlying entity (for example module, user and enrolment), while a second table stores changes in the entity (module code, e-mail address, status). This means that creating an entity actually involves inserting records into two tables, while deleting it involves marking a record in the second "changes" table as obsolete. Updating an entity requires updating the current change to mark it obsolete, then recording a new change.

These asymmetric creation, read, update and deletion (CRUD) processes are complex to

model in an object-relational mapping framework (ORM). ORM is based on the concept that the objects reflect the underlying database schema[10], which is clearly not the case here. There are a limited number of exceptions made to this rule, for example the intermediary table for many-many relationships is not generally represented by its own Java class.

Modelling data records with version control can be emulated by using two Java classes, one for each of the tables, with a one-to-many relationship used to associate them. This approach is excessively verbose, in that it uses two classes to model what is actually a single entity. Further it requires that the processes of marking records obsolete, inserting new records to approximate update operations, etc. must be implemented by the application, rather than by the framework, introducing unnecessary implementation complexity.

Extending an ORM framework to model data which changes over time would significantly reduce code size and complexity, improving maintainability. The ORM would need to support methods to load the current version of an entity, or all versions, as well as managing the asymmetric CRUD operations.

12.4.2 Merged Rendering and Parsing Definition

MMS' existing rendering and parsing frameworks were developed independently at different points in time, however on reflection there are substantial common elements required for both. Consider the requirements specified on a form input element for a fixed length string:

```
<input name="example" type="text" value="test" length="20" />
```

The "name", "type" and "length" attributes are all of relevance to the parsing framework as well, as it must understand what query parameters it should be parsing, the type of those parameters, and where relevant any length constraint. A more detailed example using the HTML 5 "number" form field type:

```
<input name="example2" type="number" value="5"
    min="0" max="10" step="1" />
```

Here the attributes “name”, “type”, “min”, “max” and “step” are of relevance to the parser. At the moment this detail is specified independently during rendering and parsing, which both duplicates effort and introduces risk of mismatches.

MMS’ rendering engine already has support for specifying details of form elements to be rendered in code, and the request parsing framework has similar tools. Finding a form of specifying the field metadata which is well suited to both use cases is the challenge in merging the two systems. Flex’s approach is to copy data into custom data structure before rendering, while the parsing framework takes in a list of expected fields. Other parsing frameworks such as JSR-303[1] use an approach based on annotating Java beans, and are likely more readily adaptable.

A rendering framework which could use JSR-303 annotated beans to render forms, would be an obvious starting point for exploring this further. MMS’ own parsing framework adds more complex rules such as verifying primary key values match a record in the database, and applying authorisation checks to entities a request refers to, which would also need to be considered as part of an effective solution.

12.4.3 Web Services

Some web services are provided by MMS already, allowing data to be presented via external systems such as the web portal or mobile applications, however these are limited and highly task-specific. Expanding these web services to provide a coherent set of services for interacting programmatically with MMS to allow for more powerful integrations would be one way of improving the system’s usefulness. This is particularly true if these web services reflect authorisation models used throughout MMS (as existing web services do), such that they can be made available to students to experiment with, or staff can use them to integrate closely with other software applications.

Example use cases could include pushing coursework deadlines directly into Google Calendar, allowing grades to be edited directly from Excel (for example using Automation[81], previously referred to as OLE Automation), direct submission of coursework from Word, etc.

[1] `http://jcp.org/aboutJava/communityprocess/final/jsr303/index.html` - accessed 4th June 2013

12.5 Features

12.5.1 Anonymisation

Future work on the anonymous marking support already present in MMS (see section 7.6.4) could look at how best to balance the contrasting requirements of anonymity against the need to be able to identify students in exceptional circumstances. For example, if a student fails to submit coursework repeatedly, or has persistent trouble with coursework, it is important that staff can identify the student, however this must be limited to such exceptions so that marking remains anonymous. This would conventionally be a matter of academic policy, and follows the idea that MMS should implement and support such policy.

From a more pragmatic point of view, there are currently issues in managing large-scale anonymisation and ensuring both that students put the correct ID on their work, and that they do not accidentally or deliberately break anonymisation. One suggested approach for managing these risks would be to encourage widespread adoption of automatic coversheet generation, which both reduces overhead in training students, and ensures the correct identifying information is attached to work.

This approach could be extended by using coversheets generated as forms (both MS Office and PDF formats support forms as part of documents, and would therefore be practical options for this), with space for markers to enter the marks. Comments could be annotated onto the document itself (this is done in many cases already), and the entire document uploaded to MMS, where the mark would be extracted from the coversheet. By ensuring the mark is attached directly to the electronic document and de-anonymisation handled by MMS, there is an opportunity to substantially reduce risk of mistakes in returning marks (for example mixing up anonymous IDs and students they relate to).

The downside to this would be that it requires marking to be done electronically, which many staff are reluctant to do so because they feel it is more difficult in comparison to marking on paper – no opinion or conclusion on this is presented as part of this thesis, however Shaw's 2008 work[119] may be of interest on this topic. It may be that e-book readers and/or tablet devices may make this more palatable for users.

12.5.2 eBook

There are a number of opportunities for the use of eBook technology in place of existing paper-heavy processes. Potential benefits include reduction in paper, automated delivery of coursework to staff, electronic recording of feedback on coursework and for that feedback to be part of a persistent digital record.

Use of eBooks poses specific challenges in document conversation from word processor formats, managing delivery to devices used by staff (and further return of content after marking). Two common formats are in use, Mobipocket for the Amazon Kindle, and ePub for most other devices; both are HTML based, however word processor formats (such as Office Open XML[2] or OpenDocument[3]) are not HTML based and require translation. Such a conversion process must be done carefully, for example to ensure no formatting issues are introduced which might affect a student's marks.

As of the time of writing, the diverse marketplace for e-readers and general lack of network connectivity makes it difficult to deliver content to them automatically, and essentially impossible to retrieve marked-up content. The Amazon Kindle has an option for documents to be delivered via e-mail, but as no equivalent functionality is apparent in other e-readers. Tablet devices have both support for user applications and readily accessible network connectivity, but their displays are perhaps less well suited to extended reading (the use of back lighting in displays in particular is frequently attributes to causing eyestrain).

Elements of this have been tackled previously, with partial success, see 7.4.1 for details.

12.5.3 MOOCs

MOOCs introduce a new style of part-time distance learning, where courses are attended by a very large number of students. However institutional staff have little, if any, direct interaction with those students. MOOCs are typically highly flexible in terms of learner participation, such that students with other time commitments can still be involved as they are able[76][77], without precluding others from being more involved where practical.

[2]Used by Microsoft Office - `http://msdn.microsoft.com/en-us/library/aa338205(v=office.12).aspx`

[3]`http://www.openoffice.org/xml/`

Potential work in this area includes attendance monitoring, enrolment management, content delivery, delivery of student work to peers for assessment, aggregating assessment grades and returning awarded grades to students. These processes are natural fits for the existing tools, although modifications would be required.

It is common for MOOCs to make use of peer assessment in marking[9][98][104][139], in order to minimise workload on academic staff. This allows the courses to scale to very large numbers of students without requiring the numbers of supporting academics which would be needed for a conventional course. However the security architecture of MMS is designed such that it is very difficult for code to allow students to see work of their peers. This is intended as a safety feature, however substantial architectural change would likely be required to support rather than hinder such a use-case.

12.5.4 Mobile/Tablet Support

While predictions of sales of tablet devices overtaking PC sales[51], have failed to materialise, mobile devices are now commonplace. Further, the introduction of low-cost tablet devices such as the Amazon Kindle HD[4], Google Nexus 7[5] and Tesco Hudl[6], all priced below £200, means these devices are now realistically within the reach of students, and frequently less expensive than conventional laptops.

Tablet and mobile devices are ideal for time-sensitive interactions with users. Examples of such interactions include:

- Delivering announcements about students' modules, such as lectures being moved or cancelled due to exceptional circumstances.
- Access to course content in lectures, such that students can follow at their own pace and refer to previous slides if needed
- Notification of course content being released.
- Group signup at start of semester. For modules with a large number of students, demand for tutorials scheduled at convenient times can be extremely high. Where

[4] `http://www.amazon.co.uk/gp/product/B00CTVTJ4Q/` - accessed 4th November 2013
[5] `http://www.google.co.uk/nexus/7/` - accessed 4th November 2013
[6] `http://www.tesco.com/direct/hudl/` - accessed 4th November 2013

these groups are available on a first come, first served basis, this leads to a significant element of time sensitivity.

- Notification of students when coursework grades/feedback, exam marks, and even final module grades are released. Delivery of the grades themselves could also be considered, however this entails substantial security risks which would need to be managed effectively.
- Recording of lecture/tutorial attendance; if the location of a lecture theatre is known (this is not trivial, as existing estates records tend to be building specific rather than room specific), as well as which lecture a student should be in, their attendance could theoretically be recorded based on presence of their mobile device in the room. There are risks of students handing mobile devices to another to "sign in" with, however it is generally accepted this already happens with existing attendance recording processes.

While MMS can be accessed via a web browser, its interface is designed for large screen devices. A move to responsive CSS (which rearranges web pages to take into account display size) or a native application could significantly improve the user experience in these cases.

Evaluation of desired functionality for mobile and tablet devices, considerations of how best to deliver that, and how to use such devices to provide functionality which may be impractical on other interfaces (for example encouraging staff to read essays in digital format by providing a hand-held device for reading from) are all potential areas for future work.

12.5.5 Email User Interface

While MMS early on was given functionality for sending e-mails (see 4.7.2) this had always been a one-way process. E-mail has been used solely to send confirmation of critical events (such as coursework submission by students) or to notify staff of changes or problems (such as students joining/leaving a module).

One area for future improvement is to look at use of e-mail not just to provide information, but also to receive it. This ties into two tools, in particular the existing coursework tool,

and a new “Journal” tool described in 12.5.6.

The motivation for the coursework tool’s e-mail interface stemmed from observing how students dealt with problems submitting work. As their main priority was to have academic staff receive the work, they tended to e-mail it directly to staff if they experienced difficulties uploading. While technical issues with MMS would be the obvious reason for this, in reality this was relatively rare, more often students would have problems with client-side hardware and software.

The key part, it was felt, was that students went to e-mail as a well understood way of dealing with the problem, and that it was a more comfortable approach to submitting work. Given that the receipt was already delivered by e-mail, in theory the exchange should be a simple matter of attaching the file to an e-mail to the right address, providing details of which piece of work it was, and receiving an e-mail to confirm the upload within a few minutes.

The first step was to arrange an e-mail account for MMS, which was at least straightforward. The account could be accessed through the IMAP protocol, allowing MMS to read in the e-mails. An API was then built to read e-mails and fire event triggers depending on the content of the mail, passing the message onto a relevant tool.

Unfortunately, the idea was essentially cut due to lack of time. Although no explicit decision was taken to abandon the feature, other requirements took priority and although in theory the API itself was functionally complete, the integration with other tools was never implemented.

12.5.6 Journal Tool

One notable gap in the functionality of MMS’ user tools is for tracking informal discussion and meetings between students and staff. This especially important for research postgraduates who may lack the more structured contact time common to taught students.

A journal tool was designed to fulfil this need, although development has been on hold indefinitely due to other tasks taking priority. The current design primarily centres on a list of events that have occurred, which each record involved parties. Events are entered by staff to keep track of what has been discussed at an informal meeting between them and

students, and may include notes, files and e-mails.

The idea of an e-mail handling system is of particular note – the intent was for MMS to have an e-mail address to which e-mail discussions are CCed. MMS would then extract the identities of the sender and recipient(s) from the email, and log it against the relevant journal. Staff could then browse a list of all e-mails that involve them, and tag them as associated with an event, or create a separate event for them.

Potential future work, beyond basic implementation, could also include logging of discussions taking place via instant messaging services, or potentially even voice over IP.

12.5.7 Server-side Scripting

One noteworthy drawback of MMS, compared to TAGS, is that it requires significantly more time to add new tools to MMS due to language choice and architecture security features. This means that development of new tools and variations on existing tools, tailored to a specific course module, is non-trivial (although some have been written).

One possible solution to this problem would be to allow server-side scripted tools. MMS already includes a Javascript scripting engine (utilising Mozilla Rhino) used for the final grade calculation tool and for internal diagnostic tools, and as such this would probably be the language of choice for such a feature. Javascript is also a relatively easy to pick up language, meaning it may be feasible for semi-technical users to add their own tools to MMS (although for security reasons such tools would have to have extremely restricted access).

If a major architectural change or rewrite was considered, a move to node.js may simplify such a design.

12.5.8 WebDAV

Management of files through a web application interface can be a slow and tedious task, while the standard desktop interface is a fast, responsive and well understood method of doing this task. One possibility that has been considered for improving the management of files within MMS is to provide a WebDAV[34] (Web-based Distributed Authoring and

Versioning) interface. This would allow the MMS "file space" to be mounted directly on a user's desktop, where for example they could copy students coursework directly to their local system by dragging and dropping the files, or even dragging and dropping an entire course's content folder in one single operation.

WebDAV functionality would fit well into the MMS core API, enabling hosted tools to expose content via WebDAV in a reasonably straight-forward manner. There are some complexities in handling issues such as location files on disk, which are not structured in the same layout as they're shown to end-users, primarily for security concerns of using user-provided names on disk, or security. Accordingly an API would need integration points for security checks and translating paths as requested by a user to locations on disk.

This could also provide an alternative administration interface, for example creating a new coursework upload slot by creating a new "directory", or even attaching a comments file to uploaded coursework by uploading it to MMS through WebDAV.

12.5.9 Blockchain-based Audit

Increased use of digital records inherently leads to challenges in managing long term archival of these records. One solution to simplifying this would be to maintain multiple copies of records, ensuring that if one is lost or damaged that multiple alternatives are available. However, in doing so care must be taken both to ensure that versions are not mixed up, and that records cannot be tampered with (i.e. altering coursework or degree grades).

A potential solution presents itself in the form of the blockchain technology used by Bitcoin. The blockchain at its core is a way to achieve distributed consensus on a sequence of events[87], and while this is conventionally used to represent the ordering of transactions, it can be adapted to prove the order (and approximate "no later than" timestamp) for existence of data.

Snapshots of data could be extracted, a suitable hash digest generated (SHA256 or the more recent SHA-3 algorithm would be suggested) and inserted into a blockchain. Similar technology is already in use for so-called Proof of Existence[7], although it is not in active

[7] `http://www.proofofexistence.com` – accessed 25th February 2015

user in higher education at the current time. The stored records could then be verified against the hash stored in the blockchain, and any tampering is evident if the hashes do not match.

Other potential uses could include decentralised coursework timestamp generation. For example where coursework cannot be submitted due to technical issues in centralised systems due to load (as described by Stoneham[124] and Parnell[101]), timestamps can be securely established for when the work was prepared and lateness penalties waived accordingly. Such use cases would create a high volume of transactions, and may not be suitable for the Bitcoin blockchain, but alternative higher-volume chains such as Dogecoin or Litecoin may be more suitable, or approaches such as Merkle-trees[79] to collate hashes into a single transaction.

Chapter 13

Conclusion

This thesis has described the research, design and deployment of MMS, a web-based application which has been shown to be effective in reducing administrative overheads at the University of St Andrews.

I have described the preceding systems and the context from which MMS was formed. Chapter 3 (Challenges) introduced the challenges inherent in developing administrative software for higher education, with further challenges in a research context. Key points here include complexity of modelling institutional administrative processes, simultaneous opposition to process changes and drivers for rapid pace of change, and difficulties in developing and supporting production software as part of a research project. Understanding the nature of challenges and how they can be addressed is important to managing software development for higher education.

Chapter 4 (Design) introduces the principles MMS is based on, taken from experiences working with TAGS and furthered through iterative design through the project lifecycle. These design principles are used to form the software architecture, integrations and tools which form MMS, and are specifically intended to address the challenges illustrated. The principles of re-using data, version control of managed data, exposing implementation detail and raising issues early are particularly important.

Having established the background and theory, the following chapters look at the actual software implemented. Chapter 5 (Software Architecture) describes the underlying framework which embodies this design and supports the user tools. This framework seeks to

abstract away much of the complexity of the underlying systems and processes, providing access to institutional data without requiring tool developers to understand the full details of the underlying systems and data structures.

This chapter also addresses the software engineering underlying MMS, showing how it achieves stability and security through defensive design and development. With regards to the points raised in the abstract, this chapter described how MMS models the complex organisational structure and authorisation roles of an academic institution (see section 5.7). Further, this architecture enables MMS' functionality to be readily and quickly modified in response to changing requirements (implementations of this are covered in chapter 7).

Chapter 6 (Integrations) looks at the external systems with which MMS integrates, for example reading student details from central records and writing back final grades. These integrations increase the value of data stored with in MMS, as well as minimising duplicated effort in managing the system. With regards to the requirements raised in the abstract, naturally this chapter relates to the need to integrate with other information systems.

Chapter 7 (User Tools) describes the main tools with which most users interact. These tools reduce administrative overheads by managing recording, auditing and reporting of key data such as attendance, marks and grades. This chapter further addresses MMS' use of server-side scripting for tools to enable end-users to inspect implementation and to rapidly modify the system's operation in response to changing requirements, addressing the rate of change challenges described above. Here the focus in tool functionality on engaging with administrative staff, in contrast to the more common learning and teaching tools which often do not consider such groups, is shown.

The following three chapters (chapter 8 through 10) examine specific case studies. The first two case studies (Academic Alerts and postgraduate research student administration) illustrate examples of tools which implement institutional policies, and the impact of use of web applications on these processes. Chapter 10 described LAVA, a tool developed by a separate research team, looking at how MMS enabled rapid development of tools by separating tool implementation from the complexities of the organisation and data they utilised.

Chapter 11 (Evaluation) assesses MMS' success against a number of disparate metrics. Primarily this consists of feedback collected through surveys and individual interviews to

determine how successful MMS has been in reducing overheads, with usage statistics used to evaluate adoption rates.

The survey results show that MMS is widely considered to have reduced overheads and improved administrative processes, meeting its intended goals and showing that a web application can indeed be used this way. Feedback from some users (especially external examiners) identifies a number of use-cases which are not as yet well supported. Interviews with highly-engaged users support the conclusions from the broader surveys, with users consistently reporting reduced time taken for administrative tasks (as well as simplification of those tasks), showing MMS' effectiveness in reducing overheads.

Through usage statistics the steadily increasing usage of MMS is demonstrated, and crucially this expansion has been primarily led by academic driven demand for the system as opposed to being a decision made centrally and imposed on academic staff. This high rate of user acceptance and growth in usage reflects positively on the functionality and usability of MMS.

Lastly, chapter 12 (Future Work) discusses a number of areas in which there is potential for MMS to be further improved, or new systems developed based on the design principles combined with lessons learnt and newer technology. There is clearly substantial scope for further expansion to the functionality of the framework, enabling it to reduce effort in developing hosted tools, particularly with regards to common tasks such as data management and user interaction.

To summarise, MMS has shown significant value over its lifecycle, and the underlying theory of providing a web-oriented software framework for improving institutional efficiency through a focus on data flow management has shown itself to be a viable and effective solution. A number of challenges to be aware of are described, and design principles outlined to address these. MMS forms a worked example of these principles and the use of web applications to reduce overheads, while there is still significant scope for future work in the area.

Bibliography

[1] Colin Allison, Alex Bain, Bin Ling, and J. Ross Nicoll. Addressing academic needs in managed learning environments. In *4th Annual LTSN-ICS*, pages 227–232, Galway, Ireland, 2003.

[2] Colin Allison, Alex Bain, Bin Ling, and Ross Nicoll. MMS: A user-centric portal for e-learning. In *DEXA '03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, pages 292–296. IEEE Computer Society, 2003.

[3] Colin Allison, Anne Campbell, Christopher John Davies, Lisa Dow, Sarah Kennedy, John Philip McCaffery, Alan Henry David Miller, Iain Angus Oliver, and Galhenage Indika Udaya Shantha Perera. Growing the use of virtual worlds in education: an opensim perspective. *Proceedings of the 2nd European Immersive Education Summit*, 2012.

[4] Colin Allison, Bin Ling, J. Ross Nicoll, David Roberts, and Luke Moodley. Addressing the disabilities information flow issue. In *Proc. 10th International Conference of European University Information Systems (EUNIS)*, Bled, Slovenia, 2004.

[5] Colin Allison, David McKechan, Hamish Lawson, and Rosa Michaelson. The tags framework for web-based learning environments. In *European Conference on Web-Based Learning Environments*. FEUP Editions, June 2000.

[6] Colin Allison, David McKechan, Alan Ruddle, and Rosa Michaelson. A group based system for group based learning. In A. Eurelings P. Dillenbourg and K. Hakkarainen, editors, *European Perspectives on Computer-supported Collaborative Learning (CSCL'2001)*, pages 43–50. Maastricht McLuhan Institute, 2001.

[7] Colin Allison, Alan Miller, Thomas Sturgeon, J. Ross Nicoll, and Indika Perera. Educationally enhanced virtual worlds. In *Frontiers in Education Conference (FIE), 2010 IEEE*, pages T4F–1–T4F–6, Oct 2010.

[8] Jo-Anne Baird and Neil Bridle. A feasibility study on anonymised marking in large-scale public examinations, 2000.

[9] Stephen P. Balfour. Assessing writing in MOOCs: Automated essay scoring and calibrated peer review™. *Research & Practice in Assessment*, 8(1):40–48, 2013.

[10] Douglas Barry and Torsten Stanienda. Solving the Java object storage problem. *Computer*, 31(11):33–40, 1998.

[11] John Batten, Jo Batey, Laura Shafe, Laura Gubby, and Phil Birch. The influence of reputation information on the assessment of undergraduate student work. *Assessment & Evaluation in Higher Education*, 38(4):417–435, 2013.

[12] Malcolm Bell and Wendy Bell. It's installed... now get on with it! looking beyond the software to the cultural change. *British Journal of Educational Technology*, 36(4):643–656, 2005.

[13] Bin Ling and Colin Allison. A User Oriented Information Portal for Learning Environment. In *Proceeding of the 9th International Conference on 'Learning technologies for communication'*, volume 2, Sunderland, UK, September 2002.

[14] Bin Ling and Colin Allison. User-centric portals for managed learning environments. In *DEXA '02: Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, pages 399–408, Washington, DC, USA, 2002. IEEE Computer Society.

[15] Clare Bradley. Sex bias in the evaluation of students. *British Journal of Social Psychology*, 23(2):147–153, 1984.

[16] Clare Bradley. Sex bias in student assessment overlooked? *Assessment & Evaluation in Higher Education*, 18(1):3–8, 1993.

[17] John Brooke. SUS - A Quick and Dirty Usability Scale. *Usability Evaluation in Industry*, 189:194, 1996.

[18] Tom Browne, Marin Jenkins, and Richard Walker. A longitudinal perspective regarding the use of VLEs by higher education institutions in the United Kingdom. *Interactive Learning Environments*, 14(2):177–192, 2006.

[19] C. O'Malley, et. al. Pedagogical Methodologies and Paradigms. `http://www.mobilearn.org/download/results/public_deliverables/MOBIlearn_D4.1_Final.pdf`, 2006. Accessed 15th October 2013.

[20] E. F. Codd. Further normalization of the data base relational model. *IBM Research Report, San Jose, California*, RJ909, 1971.

[21] Colin Allison et al. An institutionally secure integrated data environment. In *JCIEL Managed Learning Environments Circular 7/99 Executive Summary*, February 2000.

[22] Stephen Court. The use of time by academic and related staff. *Higher Education Quarterly*, 50(4):237–260, 1996.

[23] Joëlle Coutaz. PAC, an Object Oriented Model for Dialog Design. In Bullinger, H. J. and Shackel, B., editor, *Second IFIP Conference on Human-Computer Interaction*, page 431. North-Holland, September 1987.

[24] Gianpaolo Cugola and Hans-Arno Jacobsen. Using publish/subscribe middleware for mobile systems. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6:2002, 2002.

[25] Hugh C. Davis and Su White. Linking Experiences: Issues Raised Developing Linkservices for Resource Based Learning and Teaching. In *Advanced Learning Technologies, IEEE International Conference on*, pages 401–404. IEEE Computer Society, 2001.

[26] Irina Elgort. E-learning adoption: Bridging the chasm. In *Proceedings of ASCILITE*, pages 181–185, 2005.

[27] Jürgen Enders. The academic profession in europe: A view from the status perspective. `http://www.sml.hw.ac.uk/downloads/cert/wpa/1997/dp9718.pdf`, May 1997. CERT Discussion Paper No. 97/18.

[28] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, June 1999. Updated by RFC 2817.

[29] Roy Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, Computer Science, 2000. AAI9980887.

[30] Apereo Foundation. Apereo Foundation is born! Accessed 30th January 2015.

[31] Kristoffer Getchell, Alan Miller, J. Ross Nicoll, Rebecca Sweetman, and Colin Allison. Games methodologies and immersive environments for virtual fieldwork. *Learning Technologies, IEEE Transactions on*, 3(4):281–293, 2010.

[32] Kristoffer Getchell, J. Ross Nicoll, Clare Kerbey, Alan Miller, Colin Allison, Rebecca Sweetman, and Rosa Michaelson. Evaluating exploratory learning in lava. In *Proceedings of the Sixth Conference on IASTED International Conference Web-Based Education - Volume 2*, WBED'07, pages 561–567, Anaheim, CA, USA, 2007. ACTA Press.

[33] Michael Goedicke. *Paradigms of modular system development*, chapter 1. Number 17 in Computing. IEE, 1990.

[34] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen. *HTTP Extensions for Distributed Authoring – WEBDAV*, Feb 1999. Obsoleted by RFC 4918.

[35] David Goldberg. What every computer scientist should know about floating point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

[36] Jeffrey Goldberg. Why web usage statistics are (worse than) meaningless. `http://www.goldmark.org/netrants/webstats/`, 1995. Accessed 31st July 2013.

[37] Murray W. Goldberg. Student participation and progress tracking for web-based courses using WebCT. In *Proceedings of the Second International N.A. WEB Conference*, Fredericton, NB, Canada, October 1996.

[38] Murray W. Goldberg, Sasan Salari, and Paul Swoboda. World wide web-course tool: an environment for building www-based courses. *Comput. Netw. ISDN Syst.*, 28(7-11):1219–1231, 1996.

[39] Google. *chrome.pushMessaging*, 2013. Accessed 4th November 2013.

[40] Jay P. Greene, Brian Kisida, and Jonathan Mills. Administrative bloat at american universities: The real reason for high costs in higher education. Technical

report, Goldwater Institute, August 2010. http://goldwaterinstitute.org/sites/default/files/Administrative%20Bloat.pdf.

[41] Patricia J. Gumport and Brian Pusser. A case of bureaucratic accretion: Context and consequences. *The Journal of Higher Education*, 66(5):pp. 493–520, 1995.

[42] Vivek Haldar, Deepak Chandra, and Michael Franz. Dynamic taint propagation for java. In *In Proceedings of the 21st Annual Computer Security Applications Conference*, pages 303–311, 2005.

[43] Ian Harwood. When summative computer-aided assessments go wrong: disaster recovery after a major failure. *British Journal of Educational Technology*, 36(4):587–597, 2005.

[44] Christine Helliar and Rosa Michaelson. Evaluating Finesse: Experiences from the first year. In *the Scottish Meeting of the British Accounting Association, University of Stirling*, 1998.

[45] Higher education supply and demand to 2020. http://www.hepi.ac.uk/455-1907/Higher-Education-Supply-and-Demand-to-2020.html, 2011. Accessed 11 May 2013.

[46] Anthony Herrington and Jan Herrington. Authentic mobile learning in higher education. In *AARE 2007 International Educational Research Conference*. Association for Research in Education, November 2007. http://researchrepository.murdoch.edu.au/5413/.

[47] Gregory W Hislop and Heidi J.C Ellis. A study of faculty effort in online teaching. *The Internet and Higher Education*, 7(1):15 – 31, 2004.

[48] HM Treasury. Spending review 2010. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/203826/Spending_review_2010.pdf, October 2010.

[49] Wei Huang, Yao Dong, and Ana Milanova. Type-based taint analysis for java web applications. In Stefania Gnesi and Arend Rensink, editors, *Fundamental Approaches to Software Engineering*, volume 8411 of *Lecture Notes in Computer Science*, pages 140–154. Springer Berlin Heidelberg, 2014.

[50] Tim Hunt. META: enable foreign keys in the database schema. https://tracker.moodle.org/browse/MDL-30799, December 2011. Accessed 17th November 2013.

[51] Tablet shipments forecast to top total pc shipments in the fourth quarter of 2013 and annually by 2015. http://www.idc.com/getdoc.jsp?containerId=prUS24314413, Setptember 2013. Accessed 4th November 2013.

[52] Blackboard Inc. Blackboard inc. completes merger with WebCT, Inc. February 2006. Accessed 30th January 2015.

[53] Blackboard Inc. Blackboard completes acquisition of ANGEL Learning, Inc. May 2011. Accessed 30th January 2015.

[54] Ali Jafari. Putting everyone and every course online: The oncourse environment. *WebNet Journal: Internet Technologies, Applications & Issues*, 1(4):37–43, 1999.

[55] Ali Jafari. The rise of a new paradigm shift in teaching and learning. *T. H. E. Journal*, 27(3):p58–60,62,64,66,68, October 1999.

[56] JISC. INSIDE: an institutionally secure integrated data environment. Accessed 11th May 2013.

[57] Virtual learning environment activity in further education in the UK. http://www.jisc.ac.uk/uploaded_documents/VLE-in-FE.pdf, November 2003.

[58] John Browne, et. al. *Browne Report - Securing a sustainable future for higher education*. Department for Employment and Learning, October 2010.

[59] Harvey Jones and José Hiram Soltren. Facebook: Threats to privacy. *Project MAC: MIT Project on Mathematics and Computing*, 1, 2005.

[60] Mike Joy, Nathan Griffiths, and Russell Boyatt. The boss online submission and assessment system. *J. Educ. Resour. Comput.*, 5(3), September 2005.

[61] Mike Joy and Michael Luck. Effective electronic marking for on-line assessment. *SIGCSE Bull.*, 30(3):134–138, August 1998.

[62] David Kember. A reconceptualisation of the research into university academics' conceptions of teaching. *Learning and Instruction*, 7(3):255 – 275, 1997.

[63] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2004.

[64] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *ECOOP*, pages 220–242, 1997.

[65] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, August 1988.

[66] Janet Lavery, Cornelia Boldyreff, Bin Ling, and Colin Allison. Laying the foundation for web services over legacy systems. In *WSE '02: Proceedings of the Fourth International Workshop on Web Site Evolution (WSE'02)*, page 3, Washington, DC, USA, 2002. IEEE Computer Society.

[67] Janet Lavery, Cornelia Boldyreff, Bin Ling, and Colin Allison. Modelling the evolution of legacy systems to web-based systems. *J. Softw. Maint. Evol.*, 16(1-2):5–30, 2004.

[68] Avraham Leff and James T. Rayfield. Web-application development using the Model/View/Controller design pattern. In *Enterprise Distributed Object Computing Conference, 2001. EDOC '01. Proceedings. Fifth IEEE International*, pages 118–127, 2001.

[69] Larry L. Leslie and Gary Rhoades. Rising administrative costs: Seeking explanations. *The Journal of Higher Education*, 66(2):pp. 187–212, 1995.

[70] Don Libes. Writing CGI Scripts in Tcl. In *Proceedings of the Fourth Annual Tcl/Tk Workshop*, July 1996.

[71] Bin Ling, Colin Allison, J Ross Nicoll, Luke Moodley, and Dave Roberts. Disabilities Information Flow: A Disabilities Information Management System. *British Journal of Educational Technology*, 37(2):289–294, 2006.

[72] Bin Ling, Colin Allison, J Ross Nicoll, Dave Roberts, and Luke Moodley. Developing a disabilities information management system. In *5th Annual LTSN-ICS Conference*. Higher Education Academy, 2004.

[73] Pradnya Luktuke. Android application to fetch MMS result notifications. Unpublished MSc dissertation, 2012.

[74] Guangchun Luo, Yanhua Wang, Xianliang Lu, and Hong Han. A novel web application frame developed by MVC. *SIGSOFT Softw. Eng. Notes*, 28(2):7–, March 2003.

[75] Neil MacDonald and Peter Firstbrook. Designing an adaptive security architecture for protection from advanced attacks. February 2014.

[76] Jenny Mackness, Sui Mak, and Roy Williams. The ideals and reality of participating in a MOOC. In *Networked Learing Conference*, pages 266–275. University of Lancaster, 2010.

[77] Alexander McAuley, Bonnie Stewart, George Siemens, and Dave Cormier. The MOOC Model for Digital Practice, 2010.

[78] J D Mercer-Chalmers, C L Goodfellow, and G J Price. Using a VLE to enhance a foundation chemistry laboratory module. *New Directions in Teaching Physical Sciences*, 2:29–34, 2006.

[79] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer Berlin Heidelberg, 1988.

[80] Lyn Mettler. ANGEL in the classroom. September 2000. Accessed 9th March 2008.

[81] Microsoft. Automation. `http://msdn.microsoft.com/en-us/library/dt80be78.aspx`, 2013. Accessed 4th November 2013.

[82] Mobipocket. *How do I create a Mobipocket eBook*. Mobipocket.com, April 2008. `http://www.mobipocket.com/dev/article.asp?BaseFolder=prcgen&File=building.htm`.

[83] Moodle Trust. Conditional activities. `http://docs.moodle.org/25/en/Conditional_activities`. Accessed 24th October 2013.

[84] Moodle Trust. Grade calculations. `http://docs.moodle.org/25/en/Grade_calculations`. Accessed 19th October 2013.

[85] Moodle Trust. Moodle installations. http://moodle.org/sites/. Accessed 26th June 2013.

[86] Gary Motteram and Gillian Forrester. Becoming an Online Distance Learner: What can be learned from students' experiences of induction to distance programmes? *Distance Education*, 26(3):281–298, 2005.

[87] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.

[88] Melanie Newman. A funny sort of welcome. http://www.timeshighereducation.co.uk/features/a-funny-sort-of-welcome/408832.article, October 2009. Accessed 6th March 2014.

[89] Stephen E. Newstead and Ian Dennis. Blind marking and sex bias in student assessment. *Assessment & Evaluation in Higher Education*, 15(2):132–139, 1990.

[90] J. Ross Nicoll, Alan Ruddle, and Colin Allison. Redressing server limitations in network service delay. In M. Merabti, editor, *3rd Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (EPSRC PGNet 2002)*, pages 287–292. Liverpool John Moores University, 2002.

[91] David F. Noble. Digital diploma mills: The automation of higher education. *Science as Culture*, 7(3):355–368, 1998.

[92] David F. Noble. The future of the faculty in the digital diploma mill. *Academe*, 87(5):pp. 27–32, 2001.

[93] Mark Nottingham and Robert Sayre. *The Atom Syndication Format*, Dec 2005.

[94] Tomasz Nurkiewicz. Filter irrelevant stack trace lines in logs. http://www.javacodegeeks.com/2012/03/filter-irrelevant-stack-trace-lines-in.html, March 2012. Accessed 8th November 2013.

[95] NUS Connect: Anonymous marking. http://www.nusconnect.org.uk/campaigns/highereducation/archived/learning-and-teaching-hub/anonymous-marking/. Accessed 30th April 2013.

[96] Kenton O'Hara and Abigail Sellen. A comparison of reading paper and on-line documents. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, pages 335–342, New York, NY, USA, 1997. ACM.

[97] Michael O'Rourke. The development of online assessment in the Moodle virtual learning environment (VLE) as a replacement for traditional written assessment, May 2010. `https://dspace.ndlr.ie/handle/10633/5523`.

[98] Robert O'Toole. Pedagogical strategies and technologies for peer assessment in Massively Open Online Courses (MOOCs). May 2013.

[99] Cathy Owen, John Stefaniak, and Gerry Corrigan. Marking identifiable scripts: Following up student concerns. *Assessment & Evaluation in Higher Education*, 35(1):33–40, 2010.

[100] David Lorge Parnas. Software aging. In *Proceedings of the 16th international conference on Software engineering*, ICSE '94, pages 279–287, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[101] Brid-Aine Parnell. Uni plagiarism site buckles under crush of last-minute essays. *The Register*, April 2012. Accessed 4th May 2013.

[102] Indika Perera, Colin Allison, J. Ross Nicoll, and Thomas Sturgeon. Towards successful 3d virtual learning - a case study on teaching human computer interaction. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, pages 1–6, 2009.

[103] Indika Perera, Colin Allison, J. Ross Nicoll, Thomas Sturgeon, and Alan Miller. Managed learning in 3d multi user virtual environments. *International Journal of Digital Society (IJDS)*, 2010.

[104] Chris Piech, Jonathan Huang, Zhenghao Chen, Chuong B. Do, Andrew Y. Ng, and Daphne Koller. Tuned Models of Peer Assessment in MOOCs. *CoRR*, abs/1307.2579, 2013.

[105] Christoph Pimmer, Sebastian Linxen, and Urs Gröhbiel. Facebook as a learning tool? a case study on the appropriation of social network sites from mobile phones in developing countries. *British Journal of Educational Technology*, 43(5):726–738, 2012.

[106] James Pitkow. In search of reliable usage data on the WWW. *Comput. Netw. ISDN Syst.*, 29(8-13):1343–1355, September 1997.

[107] John T. Pohlmann. A description of effective college teaching in five disciplines as measured by student ratings. *Research in Higher Education*, 4(4):335–346, 1976.

[108] Dan Polawksi. Summary of violated foreign key constraints from 700+ Moodles. `https://tracker.moodle.org/browse/MDL-17623`, December 2008. Accessed 17th November 2013.

[109] David Power, Rosa Michaelson, and Colin Allison. The Finesse portfolio management facility. In *9th CTI-AFM Conference*, pages 119–125, April 1998.

[110] PwC. 2013 information security breaches survey. `https://www.pwc.co.uk/assets/pdf/cyber-security-2013-technical-report.pdf`, 2013. Accessed 16th November 2013.

[111] Eric S. Raymond. *The Cathedral & the Bazaar*. O'Reilly Media, October 1999.

[112] Eric S. Raymond. *The Art of Unix Programming*. Addison-Wesley, 1.0 edition, 2003. `http://www.catb.org/~esr/writings/taoup/html/`.

[113] Thomas C. Reeves. Storms clouds on the digital education horizon. *Journal of Computing in Higher Education*, 15(1):3–26, 2003.

[114] Hannah Richardson. Post-result university admissions urged. `http://www.bbc.co.uk/news/education-15492470`, October 2011. Accessed 8th November 2013.

[115] Chris Sangwin. Serving mathematics in a distributed e-learning environment. *MSOR Connections*, 5(2), 2005.

[116] Dirk Schneckenberg. Understanding the real barriers to technology-enhanced innovation in higher education. *Educational Research*, 51(4):411–424, 2009.

[117] Diana M. Selfa, Maya Carrillo, and Ma. Del Rocio Boone. A database and web application based on MVC architecture. In *Proceedings of the 16th International Conference on Electronics, Communications and Computers*, CONIELECOMP '06, page 48, Washington, DC, USA, 2006. IEEE Computer Society.

[118] Neil Selwyn. The use of computer technology in university teaching and learning: a critical perspective. *J. Comp. Assisted Learning*, 23(2):83–94, 2007.

[119] Stuart Shaw. Essay marking on-screen: implications for assessment validity. *E-Learning and Digital Media*, 5(3):256–274, 2008.

[120] Mark Sheehan and Judith A. Pirani. Spreading the word: Messaging and communications in higher education. 2009.

[121] Collan Simmons, Joyce Nyhof-Young, and John Bradley. Shoestring budgets, band-aids, and team work: Challenges and motivators in the development of a web-based resource for undergraduate clinical skills teaching. *Journal of Medical Internet Research*, 7(2), May 2005.

[122] Shannon Smith and Judy Caruso. The ECAR study of undergraduate students and information technology, 2010. 2010.

[123] Ray Stoneham. The impact on students of coursework assessment deadlines and penalties. In Deryn Graham, editor, *"e" Teaching and Learning 2008: Workshop Proceedings*, pages 9–13. Higher Education Academy, Subject Centre for Information and Computer Sciences, Newtownabbey, Northern Ireland, June 2008.

[124] Ray Stoneham. Coursework uploads and zero-tolerance deadlines. *Innovation in Teaching and Learning in Information and Computer Sciences*, February 2009.

[125] Dawand Sulaiman. St Andrews student driven services on iOS platform. Unpublished MSc dissertation, 2012.

[126] Symantec Corporation. *Internet Security Threat Report*, volume 18. Symantec Corporation, 2013. `http://www.symantec.com/security_response/publications/threatreport.jsp`.

[127] UK Borders Agency. Tier 2, tier 4 and tier 5 of the points based system: Appendix d - keeping documents. `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/203826/Spending_review_2010.pdf`, September 2013.

[128] Arie Van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: An annotated bibliography. *Sigplan Notices*, 35(6):26–36, June 2000.

[129] Gary Wassermann and Zhendong Su. Static detection of cross-site scripting vulnerabilities. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08, pages 171–180, New York, NY, USA, 2008. ACM.

[130] Michael Webb. MyNewport - MyLearning essentials for Facebook. In *Eleventh Institutional Web Management Workshop*, July 2007. `http://www.ukoln.ac.uk/web-focus/events/workshops/webmaster-2007/competition/submissions/#submission-9` - Accessed 6th July 2013.

[131] The University Of St. Andrews signs new WebCT licensing agreement. `http://www.bizwiz.com/bizwizwire/pressrelease/2169/848feyfex4j7wj8ke88.htm`. Accessed 15th October 2013.

[132] Karl E Weick. Educational organizations as loosely coupled systems. *Administrative science quarterly*, pages 1–19, 1976.

[133] Su White. Critical success factors for e-learning and institutional change - some organisational perspectives on campus-wide e-learning. *British Journal of Educational Technology*, 38(5):840–850, 2007.

[134] Su White and Hugh C. Davis. Capturing organisational knowledge from educational enhancement: identifying patterns for curriculum innovation. In *Enhancement and Innovation in Higher Education*, June 2013.

[135] Drew Whitelegg. Breaking the feedback loop: problems with anonymous assessment. *Planet*, (5):7–8, 2002.

[136] Pat Willmer. Proposal to establish an academic alert system. Unpublished paper to Teaching & Learning Committee, December 2009.

[137] Scott Wilson. Community-driven specifications: XCRI, SWORD, and LEAP2A. *Int. J. IT Stand. Stand. Res.*, 8(2):74–86, July 2010.

[138] Stephen Wisemen. The marking of english composition in grammar school selection. *British Journal of Educational Psychology*, 19(3):200–209, 1949.

[139] Li Yuan and Stephen Powell. MOOCs and open education: Implications for higher education. `http://publications.cetis.ac.uk/wp-content/uploads/2013/03/MOOCs-and-Open-Education.pdf`, 2013.

[140] Li Yuan, Scott Wilson, Adam Cooper, and Lorna M Campbell. The future of interoperability standards in education – system and process. July 2010.

Appendix A

Survey 2010

Module Management System

Welcome to the MMS Survey Semester 1 2009/0!

As we wrap up the first semester where the entire university has used MMS, we would like to get some feedback on how we've done, and where we should go next.

The survey is fairly short, and should take about 5-10 minutes. The survey contains three parts; demographics, usability and future planning. All questions are optional, but we would request that you complete as many as possible. Participation is entirely voluntary, and you may withdraw at any time.

The researcher for this survey is Mr James Ross Nicoll. Queries about the survey should be addressed to jrn@st-andrews.ac.uk.

Demographics

First of all, we'd like to find out a bit about yourself, so we know who we're getting responses from.

How would you describe your role within the university? If several apply, please pick the one you spend most time interacting with MMS as.

1. Prefer not to say
2. Academic (e.g. lecturer or tutor)
3. Administrative (e.g. administrator, secretary)

4. Research (e.g. research fellow)
5. Director of Teaching, Exams Officer or Head of School
6. Technical (e.g. computing officer)
7. Other

From the list below, how have you used MMS this semester?

1. Coursework handling (electronic submission and marking)
2. Coursework handling (paper hand-in with electronic marking)
3. Exam marking
4. Final module result preparation and/or submission
5. Tracking lecture and/or tutorial attendance
6. Content delivery (lecture notes, tutorial questions, etc.)
7. Tutorial group signup
8. Automated assessment (for example, the quiz tool)

If relevant, how do you feel MMS has changed the time taken for you to prepare and submit final module results?

1. Prefer not to say
2. Significantly reduced
3. Slightly reduced
4. Neither reduced or increased
5. Slightly increased
6. Significantly increased
7. Not applicable/Unsure

In generally, how do you feel MMS has changed the time taken for your department/school to prepare and submit final module results?

1. Prefer not to say
2. Significantly reduced
3. Slightly reduced

4. Neither reduced or increased
5. Slightly increased
6. Significantly increased
7. Unsure

Looking forwards to the next semester, how do you plan on using MMS?

1. Coursework handling (electronic submission and marking)
2. Coursework handling (paper hand-in with electronic marking)
3. Exam marking
4. Final module result preparation and/or submission
5. Tracking lecture and/or tutorial attendance
6. Content delivery (lecture notes, tutorial questions, etc.)
7. Tutorial group signup
8. Automated assessment (for example, the quiz tool)

Usability

Please indicate to what degree you would agree or disagree with the following 10 statements:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Future Work

Lastly, we'd like your help deciding how we prioritise tasks in the coming semester. Thinking about significant projects, please help us prioritise the following by picking your top 3 priorities:

1. Improvement to user interface design (for example simplifying pages)
2. Improved report/chart generation tools (more control over results)
3. Performance improvements (time to generate pages/charts)
4. More bulk setup options (for example setting mark-grade mapping tables across
 several modules at once)
5. Configuration copying options (copy tools and their configuration between academic years)
6. Support for groupings of modules (for example, to allow sub-honours co-ordinators to be assigned automatically to all sub-honours modules but not other modules in the same school)
7. Groupwork support in the coursework tool
8. Merging the tutorial and lecture attendance tools into a single redesigned tool
9. Windows application for students to upload work
10. Something else (answer below)

Apart from the projects listed above, are there any other features you'd like to see added to MMS?

We will be collecting more general feedback later on in the semester.

Submit

Thank you for your time!

Once you're happy with your answers, please use this button to submit the survey:

Appendix B

Survey 2013

Student Survey 2013

Module Management System

Welcome to the MMS Student Survey 2012/3!

The survey is fairly short, and should take about 5-10 minutes. The survey comprises two parts; background and usability. All questions are optional. Participation is entirely voluntary, and you may withdraw at any time. Please note that if you require support using MMS, queries should be sent to the service desk (http://www.st-andrews.ac.uk/itsupport/help/itservicedesk/).

To submit your answers, please use the submit button the bottom of this page.

Background

First of all, we'd like to find out a bit about yourself, so we know who we're getting responses from.

From the list below, how have you used MMS this academic year as a student? Please select all that apply.

- Coursework submission
- Receiving coursework marks/feedback
- Receiving exam marks
- Content delivery (lecture notes, tutorial questions, etc.)

- Tutorial group signup

How much would you say you use MMS as a student?

How much would you say you use Moodle as a student?

How much would you say you use iSaint as a student?

Looking ahead to the next academic year, would you like to use MMS on more/less modules you are taking?

Still looking ahead to the next academic year, would you like to see more or less tasks (such as coursework submission, return of exam marks, delivery of course content) make use of MMS?

Usability

Please indicate to what degree you would agree or disagree with the following 10 statements:

1. I think that I would like to use MMS frequently.
2. I found MMS unnecessarily complex.
3. I thought MMS was easy to use.
4. I think that I would need the support of a technical person to be able to use MMS.
5. I found the various functions in MMS were well integrated.
6. I thought there was too much inconsistency in MMS.
7. I would imagine that most people would learn to use MMS very quickly.
8. I found MMS very cumbersome to use.
9. I felt very confident using MMS.
10. I needed to learn a lot of things before I could get going with MMS.

Once you're happy with your answers, please use this button to submit the survey:

Staff Survey 2013

Module Management System

Welcome to the MMS Staff Survey 2012/3!

The survey is fairly short, and should take about 5-10 minutes. The survey comprises two parts; background and usability. All questions are optional. Participation is entirely voluntary, and you may withdraw at any time. Please note that if you require support using MMS, queries should be sent to the service desk (http://www.st-andrews.ac.uk/itsupport/help/itservicedesk/).

To submit your answers, please use the submit button the bottom of this page.

Background

First of all, we'd like to find out a bit about yourself, so we know who we're getting responses from. How would you describe your role within the university? If several apply, please pick the one you spend most time interacting with MMS as.

From the list below, how have you used MMS this academic year? Please select all that apply.

- Coursework handling (electronic submission and marking)
- Coursework handling (paper hand-in with electronic marking)
- Exam marking
- Final module result preparation and/or submission
- Tracking lecture and/or tutorial attendance
- Content delivery (lecture notes, tutorial questions, etc.)
- Tutorial group signup
- Automated assessment (for example, the quiz tool)

If relevant, how do you feel MMS has changed the time taken for you to prepare and submit final module results?

In generally, how do you feel MMS has changed the time taken for your department/school to prepare and submit final module results?

Looking forward to the next year, how do you plan on using MMS?

- Coursework handling (electronic submission and marking)
- Coursework handling (paper hand-in with electronic marking)
- Exam marking
- Final module result preparation and/or submission
- Tracking lecture and/or tutorial attendance

- Content delivery (lecture notes, tutorial questions, etc.)
- Tutorial group signup
- Automated assessment (for example, the quiz tool)
- Usability

Please indicate to what degree you would agree or disagree with the following 10 statements:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Once you're happy with your answers, please use this button to submit the survey:

External Survey 2013

Module Management System

Welcome to the MMS External Examiner Survey!

The survey is fairly short, and should take about 10-15 minutes. The survey comprises three parts; background, effectiveness and usability. All questions are optional. Participation is entirely voluntary, and you may withdraw at any time.

To submit your answers, please use the submit button the bottom of this page.

Background

First of all, we'd like to find out a bit about how you have used MMS. From the list below, which how you have used MMS this academic year? Please select all that apply.

- Reviewing coursework marks
- Retrieving electronically submitted coursework
- Reviewing exam marks
- Final module result examination
- Lecture and/or tutorial attendance tracking
- Reviewing course content (lecture/tutorial notes, etc.)
- Receiving exam transcripts
- Retrieving student special circumstance data

Effectiveness

This section is intended to assess MMS' effectiveness in assisting external staff. Questions will look at direct change to process, and impact on time taken.

If relevant, how do you feel MMS has affected the difficulty of acquiring marks and/or other data you require as an external examiner?

If relevant, how do you feel MMS has changed the time taken for you to act as an external examiner?

How do you feel MMS has changed the time taken for examination board meeting?

How would you feel about MMS being adopted at your own institution?

How would you feel about MMS being adopted at other institutions you are an external examiner for?

Any other feedback on MMS, in the context of your role as external examiner? This feedback may be quoted verbatim in the published PhD thesis. Please do not include details that would break anonymity.

Usability

This last section is a standardised usability survey based on the system usability scale. Please indicate to what degree you would agree or disagree with the following 10 statements:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Once you're happy with your answers, please use this button to submit the survey:

Appendix C

Interviews 2014

Interviews 2014

The interview answers provided by staff are reproduced in full below. Minor editing has been performed to match the thesis style and to ensure anonymity, and text has otherwise been left as-is.

Interviewee A

Would you describe your role in using MMS as primarily academic, administrative or technical?

Academic

Please outline the main tasks you use MMS as part of, and how you MMS in this context.

teaching: all notes and lectures are put onto MMS for the students to access them.

a. As module organiser the MMS "space" is where all the material for the module is kept and organised in a way the students are able to access easily and clearly.
b. Continuous assessment is carried out on MMS collecting the documents from students (uploads) producing turnitin scores for markers to see and access and ability to download the work for marking and upload feedback comments.
c. Access to notes about students such as disabilities, difficulties, illnesses, absences etc.
d. Emailing groups or classes is easy

Overall, how would you say MMS affects the time taken for these tasks?

MMS enables the tasks to be efficiently carried out in an interactive way. The time taken for each task is cut by various MMS tools such as ordering students by ID or by name or by assessment. Missing assessments can be identified quickly and high turnitin scores at a glance.

How does MMS, if relevant, assist with these tasks?

MMS is interactive. All members of the staff team can access the information and add to it where necessary any time.

How does MMS, if relevant, hinder these tasks?

Sometimes if MMS module sites become cluttered by poor management then finding important information could involve opening many windows and the information might be within a folder, within another folder within another folder etc which takes time to get to.

In your opinion what are the best and worst aspects of MMS?

a. best – access to all the information by staff and students.

b. worst – finding items can be tricky if they are in sub sub folders

What would you want to change about MMS?

Is it possible to include an index? So one can click on the correct item right away? (this is for "content". Otherwise I like the way it works for management of continuous assessments and exams.

Exam suggestion: on xl sheets one can use the up down keys to add marks this is not possible in MMS so adding marks on the system is not usually carried out. We usually download the xl sheet and upload it again once marks are entered. A closer to xl like function for that would be great!

Email system: cannot do underlines or bold or any font changes which would sometimes be nice.

Any other comments on MMS?

I have arrived at St Andrews just as it was beginning and it is much better than the previous system. Especially during exam times it makes management of all exam marking so much better. I like using it for class material and find it easy to arrange the material for the students.

Giving feedback is easy although downloading documents and uploading them again is time consuming and it would be great if we could write comments on the document without doing that. Basically it is a great tool for classroom management.

Interviewee B

Interviewee did not provide a response to the interview questions

Interviewee C

Would you describe your role in using MMS as primarily academic, administrative or technical?

My role is administrative with very occasional technical work

Please outline the main tasks you use MMS as part of, and how you MMS in this context.

- Initial activation of modules each semester
- Allocation of Staff to modules (including Externals and PG Tutors), hence allowing correct people to have access to correct modules.
- Ensuring Evaluations are correctly set up for staff and PG Tutors on modules
- Receipt of module booklets from academic staff, checking for errors and uploading to MMS
- Initial configuration of modules as per individual module booklets. Where a configuration has been carried over from a previous manifestation of a module, amending the configuration for new dates or changes to methods of assessment, as per the module booklet.
- Sometimes set up Tutorial Groups and other specialised functions for staff.
- Enter notes per student per module if special circumstances need to be drawn to staff attention
- Bulk e-mailing of students per module and even per tutorial group.
- Occasional upload of late work for students.
- Deal with waivers of penalties, as per instructions from academic staff (usually bulk requests once per semester across all modules, following exam board meetings)
- Monitor all deferred assessments and remind academic staff of outstanding tasks.
- Monitoring of "problem" students across all modules that they are taking.
- Do all the preparations/corrections to module marks before Exam Board meetings, so that results are ready to upload once results are approved (NB, I do NOT do the upload itself, as this is reserved to Exams Officers).
- Liaise with academic staff and/or MMS team about anomalies that need solutions

Overall, how would you say MMS affects the time taken for these tasks? Mostly has made my life a lot easier and stopped me having to stare endlessly at spreadsheets. Also, staff can enter marks themselves,

instead of me having to nag to get them – this also reduces any possible problems of recording marks through miscommunication.

How does MMS, if relevant, assist with these tasks?

- More or less obliterates the need for spreadsheets and the inherent possibilities of getting formulae wrong, hence messing up the end results for students and the module (still need the occasional spreadsheet but this can be downloaded from MMS and, if need be, adapted to a particular member of staff's requirements).
- Makes access to all material for a module much easier for all concerned, staff and students alike.
- Makes keeping track of what is going on in a particular module much easier, as well as keeping track of all modules across the School.
- Makes keeping track of extensions/penalty waivers etc easier
- Makes keeping track of Deferred Assessments much easier

How does MMS, if relevant, hinder these tasks?

- Occasionally a bit clunky
- Can sometimes be awkward if a multi-week, multi-choice assessments are devised by academic staff. This is when assistance is usually sought from MMS team to write a special configuration.

In your opinion what are the best and worst aspects of MMS?

- It can occasionally be rather clunky/clumsy/frustrating because it doesn't quite do what you expect. However, with practice this gets easier
- Not always intuitive, although once you are used to its quirks, you know how to work your way around it
- Bad that MMS guides are not updated regularly as new functions are written into the programme – the same guides on the initial Log In page are there from when it was first introduced and have not been updated!

What would you want to change about MMS?

Not sure – maybe do bulk upload of module booklets via School View to multiple modules at once (i.e. tick boxes next to a module in a list and be able to attach the relevant PDF module booklet. In my School this would save me having to go individually into 27 to 30 modules per semester, which is very time-consuming)

Any other comments on MMS?

No, other than I do like using it, as it is pretty straight forward.

Interviewee D

Would you describe your role in using MMS as primarily academic, administrative or technical?

As School Administrator, I am involved in working with all aspects of MMS in an all-round administrative capacity (Unit/Module Administrator role for all School modules).

Please outline the main tasks you use MMS as part of, and how you use MMS in this context.

- Monitoring the numbers of students who are being advised into a module as they filter into MMS
- Setting up tutorial/laboratory sessions for attendance recording (times, dates, weeks)
- Setting up sign-up options for students to select suitable tutorial/laboratory groups that fits with their timetable
- Monitoring tutorial/laboratory attendance (tutors complete tutorial attendance themselves)
- Using information from above to issue, if necessary, Academic Alerts
- Accessing submitted self-cert forms to check for validity of absence (to tie in with compulsory tutorial/laboratory attendance)
- Accessing Disabilities Information with regard to any specific requirements in relation to class test and examination provision (I am also the School's Disabilities Co-ordinator)
- Monitoring Turnitin reports and alerting Academic Misconduct Officer to potential plagiarism issues in submitted continuous assessment
- Issue Academic Alerts as and when required
- Setting up Continuous Assessment tool for students to submit written work and marks to be recorded (in my School, as Administrator, I record all continuous assessment marks – I don't trust academic staff to do this correctly or to double check with what hard copy is being given back to the students).
- Setting up Final Grade tool for calculation of overall module grades including setting models for continuous assessment and examination weighting).
- In conjunction with the School's Examination Officer, uploading of student final module grades directly to Registry
- Assign Staff to modules as required and indicating those who should be assessed as part of CAPOD's Module Evaluation

Overall, how would you say MMS affects the time taken for these tasks?

- Before MMS, all continuous assessment and examination marks had to be recorded on Excel spreadsheets. Statistical data (average, standard deviation, range) etc used to have to be done manually, which was very stressful and open to error, particularly with the time constraints near to reporting

and degree classification times. MMS produces such data and it will also produce certain data graphs illustrating results.

- Before MMS, it was necessary to use E-Vision to access submitted Self-cert forms and Disabilities information. This has been brought easily together in MMS to make retrieval more convenient and less time-consuming.

How does MMS, if relevant, assist with these tasks?

The fact that MMS accurately calculates data mentioned above saves a lot of time and reduces the margin for human error.

How does MMS, if relevant, hinder these tasks?

I can't think of any way in which MMS hinders these tasks.

In your opinion what are the best and worst aspects of MMS?

Best:

- Ease of use
- Ease of monitoring student's tutorial/laboratory groups and identifying where student are at any given time
- Penalties for late submission of work are automatically calculated
- Turnitin availability

Worst:

- Content section is not as inviting as WebCT was of as Moodle is now
- The system can slow up at times when busy

What would you want to change about MMS?

- In Tutorial Attendance page, I'd add a sort function to the final column "Unapproved Absences". This was essential when dealing with a module of some 380+ students and you need to view numbers of absences that could lead to the issue of an Academic Alert.
- In Tutorial Group tab at the top, after scrolling down to see the list of students alphabetically with the selected dot in their selected group, again in a very large module with 380+ students and a choice of 24 separate groups (eg. EC1002) it is very difficult to change a student's group without have a freeze frame option for the group numbers.

- Following reporting of August Deferred and Reassessments results, I would like these to be retained in MMS (as do the December and May results) for future reference. Currently they disappear following reporting.
- I would like to see the Content Section become a bit more user-friendly and inviting for the students to use – more like Moodle perhaps if possible.

Any other comments on MMS?

I'm glad the system was introduced as it makes many things more efficient and keeps track of information in a much more efficient way than what was available to me before its introduction.

I believe it is something that has the capacity to evolve and grow as required.

Interviewee E

Would you describe your role in using MMS as primarily academic, administrative or technical?

Administrative

Please outline the main tasks you use MMS as part of, and how you MMS in this context.

- Tutorial organisation
- Attendance
- Academic Alerts
- Disabilities
- Coursework Marking & Final grading
- Uploading to Registry
- CSV files for Exam Boards

Overall, how would you say MMS affects the time taken for these tasks?

It reduces the need for paperwork immensely – the system has significantly reduced the hours required for each individual task.

How does MMS, if relevant, assist with these tasks?

It keeps all the tasks in one place and the links and updates across fields if helpful. It is easy to use and produces reports if required.

How does MMS, if relevant, hinder these tasks?

It doesn't – the only hinderance is human but that can be overcome by training.

In your opinion what are the best and worst aspects of MMS?

Best: Ease of use, quality of data, reports and uploads available, works across departments and units, access is easily given to other members of staff. Worst: Roll over each year of modules seems to be fraught and there are some errors even when the correct information is requested well in advance.

What would you want to change about MMS?

Just the roll over each year.

Any other comments on MMS?

I know that MMS has its critics in each school or unit but I suggest that is because of lack of training or regular use. It has saved our office time and effort and as I say above, it keeps everything in one place and is easily managed by course administrators with the correct level of access available for each member of staff.

Interviewee F

Would you describe your role in using MMS as primarily academic, administrative or technical?

Primarily academic – as a lecturer on a range of modules, across ugrad and PGT levels.

Please outline the main tasks you use MMS as part of, and how you use MMS in this context.

- For level one ugrad – delivery (particularly timed release) of handouts, repository of digitised extracts prepared by Library, tutorial sign-up, tutorial attendance monitoring (via PG tutors keeping register of attendance at the tutorials they lead), assignment upload and return of feedback, uploading of marks to Registry, emailing module for whole class messages.
- For honours ugrad level – as above.
- For pgt level, in this case distance learning – as above expect for tutorial attendance registration.

Overall, how would you say MMS affects the time taken for these tasks?

Management of tutorial group assignment takes a very small amount of time now in comparison to allocations by paper sign-up sheets in a class; monitoring of attendance much easier as pg tutors directly input attendance. Recording of marks – especially for distance learning pgt modules is much less time consuming than previous use of an excel spreadsheet. When different markers are giving feedback and moderating involved this is also much more efficient in terms of time because all materials are accessible for review and one-click release of marks to a class is so much quicker than sending individual emails (and less prone to error).

How does MMS, if relevant, assist with these tasks? See above responses.

How does MMS, if relevant, hinder these tasks?

The feedback tool has never been as intuitive as other tools in the system. The amount of clicking through to review feedback when moderating is cumbersome – and I believe annoying for our external markers. It's always been difficult to be sure what students see in terms of the history of uploading marks and feedback – so if feedback requires to be amended following moderation, and/or a second marker's comments are required this takes extra time – especially as overwriting / supplementation of previous feedback is not transparent to users.

In your opinion what are the best and worst aspects of MMS?

Best is the tutorial sign-up tool. Worst is the feedback tool (for the reasons stated above). Options for adding notes on individual students - that don't then screw up the print out of a grid of marks comes a close second for worst aspect.

What would you want to change about MMS?

Feedback tool would be much more transparent so that it clearly shows staff what's (going to be) visible to students. Option for adding notes needs to be suppressible when it comes to printing a grid of marks.

Any other comments on MMS?

It's a vast improvement on the manual system I remember. I've used the coursework tool within the PGR module in order to manage students' travel grant applications (which was messy but worked). For a School with over 70 pgr students it would have been great if MMS had been supported institutionally to the extent that additional functionality for these, and perhaps other, administrative processes could have been refined.

Interviewee G

Interviewee did not provide a response to the interview questions

Interviewee H

Would you describe your role in using MMS as primarily academic, administrative or technical?

Administrative.

Please outline the main tasks you use MMS as part of, and how you use MMS in this context.

- Staff – to update the list of staff with roles.
- Content – the upload of module and programme handbooks.

- Coursework – to create upload slots for coursework, and then to enter marks and feedback.
- Exams – to create an exam tool for the upload of exam marks, and to upload marks.
- Final grade tools – the creation and use of final grade tools.
- Additional exam and final grade tools for deferred and resit marks and grades.

Overall, how would you say MMS affects the time taken for these tasks?

It is more efficient (and accurate) than using excel spreadsheets, as there is less room for error, especially as we use the MMS option to enter marks twice. The final grades are calculated for us without the need for complicated excel formulae.

How does MMS, if relevant, assist with these tasks?

As above.

How does MMS, if relevant, hinder these tasks?

It is not able to cope with complicated modules (for example, the *[removed]* suite of modules, with different credit weightings, where students may choose from a number of Units and submit a selection of assignments).

Deferred and resit results sometimes seem to "disappear" from MMS and are not available for previous years. The module record for all students is incomplete.

In your opinion what are the best and worst aspects of MMS?

Best – it is centrally stored and backed-up, giving peace of mind.

Worst – it has limitations – see items 5 and 7.

What would you want to change about MMS?

In our first year modules *[removed]* we give the students a choice of 6 essays from which they should choose, write and submit one essay. The different essays have different submission dates (and different seminar dates), but MMS is not able to assign different submission dates to different groups of students. Instead, I enter the due date in the MMS coursework tool as the date of the last essay. Despite constant reminders, some of the students are still confused and miss their specific deadline dates, citing MMS and iSaint as having given them their (incorrect) deadline information. I would like to be able to set up MMS so that every student has their own individual essay deadline date showing, within the one coursework tool.

Any other comments on MMS?

MMS should be an electronic resource for students where they can look back over previous semesters and years and view their coursework submissions and the electronic feedback given. However, I understand that students are not able to view the modules that they have taken in previous years. This means that they are obliged to take electronic or paper copies of their work and feedback to be able to refer to in in later years.

Interviewee I

Would you describe your role in using MMS as primarily academic, administrative or technical?

Technical with administrative

Please outline the main tasks you use MMS as part of, and how you MMS in this context.

- I use MMS when setting up Moodle courses, which I do for the start of each semester or as required for non-semester-based courses such as for the School's "Executive education" programmes.
- I set up access for Moodle – such as Read Only for certain non-lecturing academics – via MMS.
- Prior to this year, there were scanned teaching materials on MMS that I checked.

Overall, how would you say MMS affects the time taken for these tasks?

Re setting up Moodle courses. As MMS is the current interface for the job, and was developed concurrently with Moodle for this, I can't make a comparison with another way to do the task. The previous bought-in VLE WebCT did or was able to do much of what our own MMS & Moodle do. Re provision of scanned teaching materials, I suspect that MMS was standing in until Moodle became acceptable for the purpose as it is now via the Library's Reading List tools which are a bit more straightforward.

How does MMS, if relevant, assist with these tasks?

My main use of MMS is in service of Moodle provision of online teaching/learning materials.

How does MMS, if relevant, hinder these tasks?

I do not experience MMS as a hindrance.

In your opinion what are the best and worst aspects of MMS?

The development and maintenance teams for MMS/Moodle have been thoroughly professional and responsive to user requirements and concerns. The awkward thing with any online tool is that if there are problems with servers or other elements of access, the work cannot be done.

What would you want to change about MMS?

Any additional documentation would be good for a user like me – how to do things and why. This is an area that has seen good development. (I am being encouraging rather than negative!) Also, the more tracking the better – who did what and when.

Any other comments on MMS?

I know that other users have found the tool increasingly useful, if a bit frustrating at times of high demand/slow functioning/non-availability.

Interviewee J

Would you describe your role in using MMS as primarily academic, administrative or technical?

Academic.

Please outline the main tasks you use MMS as part of, and how you use MMS in this context.

As a lecturer, I primarily use MMS for document delivery (lecture notes, problem sheet solutions, generic feedback documents on work done) and tutorial group signup for students registered on my module.

As director of teaching, I use a variety of systems within MMS: recording which staff and students are to be evaluated using the University's module evaluation questionnaire system, adding staff and students to modules (primarily, for me, for the evaluation questionnaires), and the degree classification system.

(Although I have authorisation to use the grade reporting system as DoT, it is our examination officer who mainly uses that.)

Overall, how would you say MMS affects the time taken for these tasks?

On the whole, I find most parts of MMS that I use to be reasonably rapid to use. Document delivery works pretty quickly for upload – the only part that I find slightly cumbersome is the need to switch to table format when providing more detailed comments to supplement file names.

Tutorial group signup works quite well, though I sometimes find that I need to hunt for a while to find the right place to configure them. (The fact that I only see "Configure" when I go into the Tutorial Signup tool, but not when I click on the Tutorial Groups tab seems surprising to me.)

Setting up module evaluations seems a lengthy process whenever I do it. On the other hand, I cannot myself envisage a way that this could be set up in a more streamlined way myself.

The Degree Classification system is pretty transparent. It is certainly easier to use than the old system. The only things that slows the process down is that there remain quite a few bugs in the system (mostly in rounding of calculated means) but more significant is battling with data incorrectly held by Registry.

How does MMS, if relevant, assist with these tasks?

Really quite well. Document delivery and tutorial signup both work very smoothly.

The DoT tasks are more functional but on the whole MMS assists quite well. The difficulties for them I've already outlined in my answer to question #3.

How does MMS, if relevant, hinder these tasks?

As in #3, the main issue I feel is bugs with the degree classification system.

In your opinion what are the best and worst aspects of MMS?

Best aspects: the uniform system that is reasonably straightforward to use and can be used across my parts of university teaching.

Worst aspect: outstanding bugs that do not seem to have been addressed.

What would you want to change about MMS?

I have recently discovered that it is possible to create some resembling a module, but not actually corresponding to something bearing a University module code. We used this to create an area for a separate computing workshop running in parallel to one suite of modules. The issue, however, is that we needed the MMS team to create this for us. It would have been nice to have the power myself to create this.

Any other comments on MMS?

On the whole I find it a rather good system. It seems much easier to use than the VLE systems for setting up areas for module document delivery.

Interviewee K

Would you describe your role in using MMS as primarily academic, administrative or technical?

administrative

Please outline the main tasks you use MMS as part of, and how you MMS in this context.

I use MMS to enable students to:

- Submit their coursework
- Receive their coursework marks
- Sign up for tutorial groups
- Sign up for group work

Additionally I use MMS to:

- Record all coursework and exam grades
- E-mail students either in groups or individually
- Calculate final module grades and create spreadsheets for Exam Boards
- Check Disability arrangements for class tests and exams

- Record and monitor tutorial attendance from class registers
- Assist Examinations Officer with module grade reporting to Registry

Overall, how would you say MMS affects the time taken for these tasks?

MMS greatly reduces the time taken to complete these tasks.

How does MMS, if relevant, assist with these tasks?

MMS provides an easy-to-use platform to maintain student records which is visible to all those with access to the information

How does MMS, if relevant, hinder these tasks?

No evidence of any hinder in completing these tasks.

In your opinion what are the best and worst aspects of MMS?

The best aspects are:

- the interface MMS provides between staff and students. Students can easily choose tutorial groups and make changes up to the completion deadline.
- Monitoring submission of coursework is very valuable as are the automatic Turnitin checks.
- The ease and reliability of making calculations in arriving at module grades.
- Being able to e-mail whole module classes or individual groups

What would you want to change about MMS?

- Downloading into pdf form can often look clumsy with large modules, particularly when student names are divided up
- The choice of "group types" being placed at the bottom of long student lists is not the in the easiest place to find. Perhaps this could be on a group signing up configuration page.

Any other comments on MMS?

MMS has greatly improved the way administrative staff carry out routine but important tasks. The ability of staff to view and/or mark student assignments away from St Andrews is hugely beneficial.

It has been one of the best systems innovations in St Andrews.