Feedback Loops in Educational Environments using Web-Based Survey Tools: New Technology Development and Three Implementation Case Studies

by

Benjamin Spead

B.S.E.E.
Grove City College, 2001

Submitted to the Engineering Systems Division and the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirement for the Degrees of

Master of Science in Technology and Policy
and
Master of Science in Electrical Engineering and Computer Science

at the Massachusetts Institute Of Technology

June, 2004

Author _____
Technology and Policy Program, Engineering Systems Division
Department of Electrical Engineering and Computer Science
May 13, 2004

Certified By _____
Dr. Joel Cutcher-Gershenfeld
Senior Research Scientist, Sloan School of Management
Thesis Supervisor

Certified By _____
Dr. Stuart Madnick
John Norris Maguire Professor of Information Technologies
Thesis Reader

Accepted By _____
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Accepted By _____
Dava Newman
Associate Professor of Aeronautics and Astronautics and Engineering Systems
Director, Technology and Policy Program

Feedback Loops in Educational Environments using Web-Based Survey Tools: New Technology Development and Three Implementation Case Studies

by
Benjamin Spead

Submitted to the Engineering Systems Division and the
Department of Electrical Engineering and Computer Science on May 13, 2004

in Partial Fulfillment of the Requirements for
the Degrees of Master of Science in Technology and Policy and
Master of Science in Electrical Engineering and Computer Science

ABSTRACT

This thesis presents lessons from the development of an on-line, web-based feedback system and preliminary analysis of the socio-technical interactions associated with the specification, design and use of this system. The system has been designed to facilitate many forms of feedback in educational environments. Novel technical components of the system include: (1) a highly detailed and flexible data model representing the social-system feedback loops, (2) reusable feedback objects and sub-objects which minimize content entry requirements while improving data aggregation capabilities, and (3) feedback objects which respond dynamically based on the context in which they are used. Novel social system learning associated with this thesis centers on the mechanisms to facilitate feedback in academic and industry learning environments.

This thesis features the design, implementation, and preliminary use of the Online Feedback Organization, Requesting and Monitoring for Educators (OnFORME) system. OnFORME was created to enable ubiquitous feedback collection in a semi-transparent manner for educators in both academic and corporate training environments. OnFORME provides a way for educators to quickly author feedback objects (e.g. surveys) and publish them via the World Wide Web so that students can easily respond. The author used agile development methodology to design and implement this application directly from customer use cases in a bottom-up approach.

Inductive research methods included observations of thirteen potential adoption trials, of which nine progressed to full usage trials. In depth interviews of potential users, actual users, and experts in this field led to insights on the adoption and use of new IT systems in education. Findings based on usage trials include (1) adoption is more likely when a parallel set of supporting institutional arrangements already exists, as well as specific feedback tools already in use, (2) face-time and hands-on demonstration are required for new system adoption, (3) including feedback as part of a predefined student role improves response rates, and (4) performing a training survey prior to the students' departure from a classroom setting increases the response rate.

Thesis Supervisor: Dr. Joel Cutcher-Gershenfeld
Title: Senior Research Scientist, Sloan School of Management

Thesis Reader: Dr. Stuart Madnick
Title: John Norris Maguire Professor of Information Technologies

Contents

List of Figures

Part I – Overview

1 Introduction

Developing effective mechanisms for feedback collection in learning environments is particularly important at the frontiers of new knowledge. This thesis presents the learning associated with designing and implementing OnFORME[1], a web-based application intended to support feedback collection in educational environments. This thesis also presents the learning associated with trials of the OnFORME system in a variety of educational settings (within both academic and corporate training environments). The overarching motivating question being addressed is:

> Can a dynamic feedback system be designed and implemented to enable semi-transparent, ubiquitous feedback collection options for educators?

- By "semi-transparent" I mean that the system will require minimum interaction with users, and those interactions that are required will involve a minimum amount of effort on the part of those interacting with the system.
- By "ubiquitous", I mean that the system will enable feedback collection in as wide a variety of forms and venues as possible (e.g. in-class evaluation, weekly feedback, spot-checks on content comprehension, etc).

The technical investment was made with the goal of completing a production-quality release of a software application. As the sole technical investigator on this project, it was imperative that I deliver a fully functional piece of software prior to graduation; else the efforts would have most certainly ceased following my departure. Having delivered such a finished system, the intention is that it will generate enough buy-in from trial users that they would be willing to invest in the project and keep it moving forward. Plans to enable this are also included in this thesis.

The system usage experimentation invested in this project represents exploratory inductive research designed to generate hypotheses, rather than deductive hypothesis testing. The intention is that further work be done (using the fully-functional technical output of this work) to deductively investigate the social-use suppositions that are tentatively presented in this thesis.

1.1 Illustrative Example

In order to illustrate how such a system could be used in practice, the following is an example of an actual use case. This case has been created based on an interview with a future user, Paula Sparling (P. Sparling, Personal Interview, March 17th, 2004). The case includes time estimates given by Paula based on her current process, and compares them with anticipated time estimates based on using the functionality of the OnFORME

[1] OnFORME stands for Online Feedback Organization, Requesting and Monitoring for Educators.

system. Comparing the two processes reveals an anticipated time reduction of 92% in Paula's feedback collection processes.[2]

1.1.1 Background and Example Motivation

Paula Sparling is the Course Administrator for the MPhil in Technology Policy (TP) at Cambridge University, U.K. Her responsibilities include conducting a student survey at the end of each term for each module[3] taken that term by TP students. The survey is used to collect the students' opinions on each module's content and the module instructors (or "lecturers"). In this particular example, Paula was doing surveys at the end of Lent term (January through March), and each student in the TP program needed to do 5 surveys, aligning with the 5 modules that had been taken during the term. All of the surveys are anonymous, and they comprise the front of a single page. The form distributed by Paula to the students is shown in Figure 1-1.

[2] These are projected time reductions. It was not possible to verify this estimation prior to the completion of this thesis. However, the other use cases indicate that these time estimations are fair projections.
[3] A "Module" at Cambridge University is an eight-week series of lectures, which might be thought of as a "Course" in the U.S. educational system.

JUDGE INSTITUTE OF MANAGEMENT STUDIES

University of Cambridge

MPhil Individual Course Feedback –Lent Term 2004

Please insert a number in each box between 1 and 5, where 1 indicates very poor and 5 indicates very good.

COURSE									
LECTURE(S)		Person A	Person B	Person C					
	Usefulness								
	Interest								
	Clarity								
	Handouts								
	Difficulty		1 = v.easy,				5 = v. difficult		
	Speed		1 = too slow				5 = too fast		
Specific comments on the course									
What were the best things about this course?									
What were the worst things about this course?									
What improvements would you suggest for the course?									

Figure 1-1: Example Survey used by the TP MPhil Program at Cambridge University[4]

[4] NOTE: This example survey is a couple of months out of date. Paula informed me that the new survey does not have the word "Studies" in the title, and that it explicitly states that it if for the "MPhil in Technology Policy".

The survey shown in Figure 1-1 is an example for a module with three lecturers: PersonA, PersonB, and PersonC.[5] For the Lent Term evaluation process, Paula reported that she performed a series of steps, and she estimated how much time she spent on each step. This data is shown in Figure 1-2.

Task	Time (hrs)
Photocopy survey, email students, and chase them to come and take the survey	2
Sit with them while they take the survey	1
Chase those who didn't come in and have them take it[6]	2
Manually enter all the survey response data into a spreadsheet	8
Collate information, photocopy all of the responses for each lecturer and distribute	3
Chase lecturers to get comments	2
Compile lecturer comments	3
Make final report	3
TOTAL:	**24**

Figure 1-2: Current Tasks and Duration for the TP Surveys

It took Paula approximately 24 hours (or two thirds of one work week) to do an end of term evaluation process on five modules, attempting to collect results from 27 students (and in the end only 26 submitted the forms).

Suppose Paula wished to replace her paper-based system with a digital system that could automate most of the work for her. Using the functionality currently available through the OnFORME system, the tasks and times required for her to obtain the same results would be reduced to those listed in Figure 1-3.[7]

[5] When the survey is actually used, the students sometimes insert the names of the instructors, or some choose to insert the lecture numbers, or some just do a general evaluation. The lack of uniformity further complicates the collection process.

[6] Even though the surveys are anonymous, Paula must make sure that each student submits all of their surveys, and so she must manually keep track of which students have submitted which surveys.

[7] These numbers assume that OnFORME is being used in the steady state, after the associated learning curves have been climbed (as they have been with the current system).

Task	Time (hrs)
Create the courses, add the students and surveys to database[8]	0.5
Email the students with links to the Surveys[9]	0
Email those a week later who have not completed the survey[10]	0.5
Chase lecturers to get comments[11]	0.5
Print the final report	0.5
TOTAL:	**2**

Figure 1-3: Tasks and Duration for the TP Surveys using OnFORME

Comparing the total number of hours required by each approach shows an estimated time saving of 22 hours, or a 92% improvement in efficiency. The details of how OnFORME enables such a substantial process improvement is central to understanding this motivating example, and so the following sections detail the exact processes that Paula would follow.

1.1.2 Preexisting Objects in the Database

This exercise assumes certain things about the state of the OnFORME system prior to the work Paula would do to collect survey responses at the end of the term.

1) It is assumed that the surveys used have previously been used in OnFORME, and therefore they do not need to be created from scratch (in the same way that the paper survey does not have to be rewritten each time it is used). If the survey did have to be created, it would not be an arduous task, assuming that the content had already been decided upon.[12,13]
2) It is assumed that the modules taught during the term have been previously taught and that they therefore have Subjects defined for them in the system.[14] If the Subjects need to be added, they would take about 60 seconds each to create.
3) It is assumed that the students are already in the database, which would happen when the masters program begins. The students have the option of entering their own information, which decreases the administrative overhead of the system.

[8] This time estimate is based on the assumption that the MPhil Survey has previously been entered into OnFORME, which means that it can then be reused in as many different courses as desirable.
[9] This task takes place in a matter of minutes using the built-in communication features of OnFORME, so it is listed for task accuracy, but it does not significantly add to the total task time.
[10] This chasing time could be further reduced and possibly eliminated by making the release of the students' marks contingent upon the submission of the surveys. This is currently being considered as an incentive for responding.
[11] This assumes that the instructors will log into the system and submit their responses digitally.
[12] In this actual use-case, creating the paper survey in Figure 1-1 within OnFORM took about 30 minutes, and that task will never have to be repeated again.
[13] Making the assumption of the surveys already existing in the database skips the steps involved in creating a Survey in OnFORME. The survey creation process is central to the feedback collection process, but tangent to this example. When the same survey is going to be reused across many courses or modules (as is the case in this example), creating the survey is a one-time process. Further details on the particulars of the survey creation interfaces can be found (to a small degree) later in this thesis, and in-depth tutorials are available in the OnFORME online help files.
[14] Later in the thesis the objects in the educational structures such as Subjects will be more fully discussed. For now, assume that the Subject is a general instance of a Course or module.

4) It is assumed that the instructors already have profiles within the system.

With a stable process in place using OnFORME for the MPhil evaluations, these assumptions will likely be true. If not, their one-time overhead cost is minimal compared to the overall time savings.

The final result of making a digital version of the paper-based survey (shown in Figure 1-1), is presented in Figure 1-4.

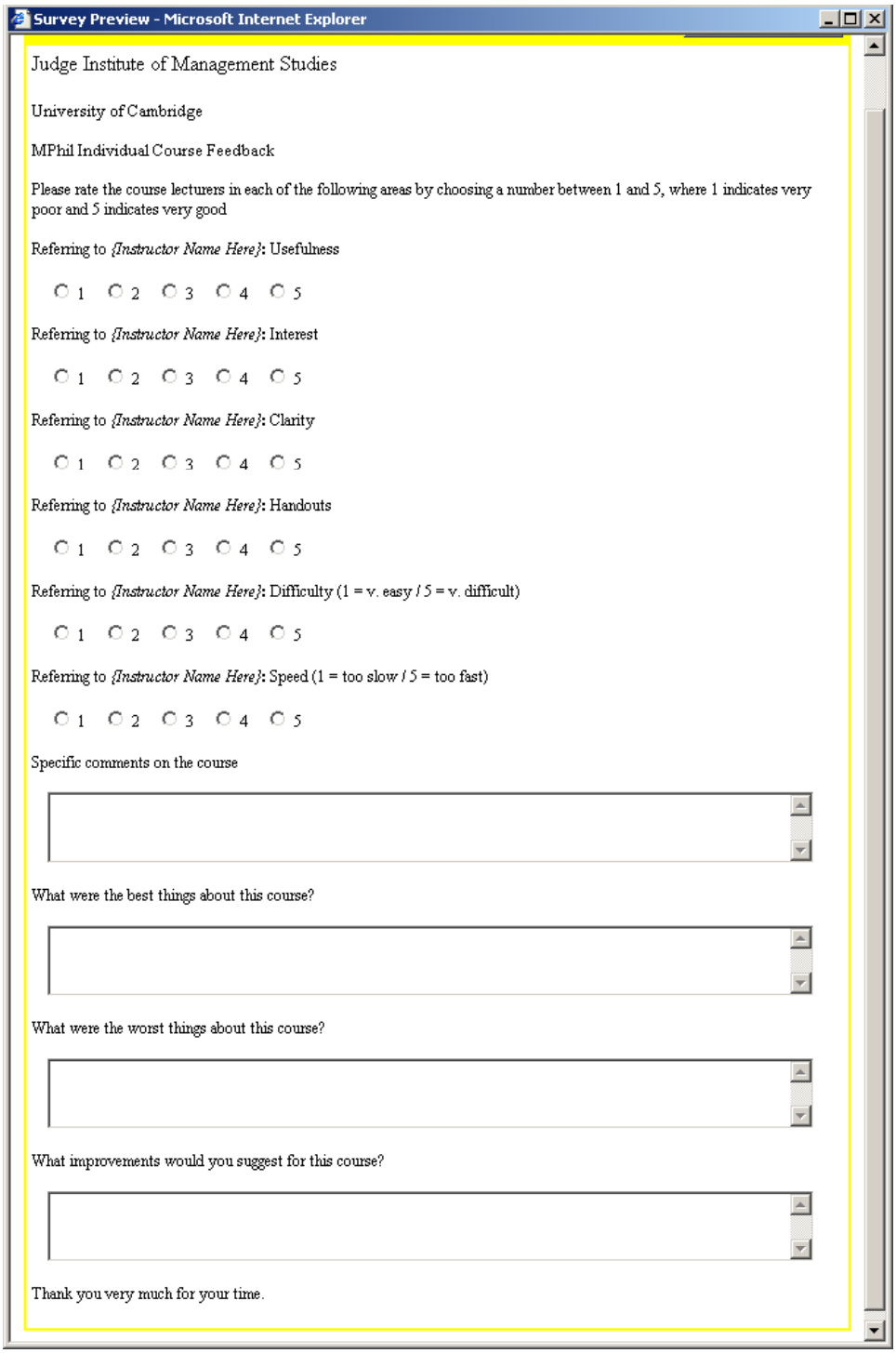


Figure 1-4: MPhil Survey Preview in OnFORME[15]

Given the above assumptions about the initial state of the OnFORME system, the following steps would be taken to collect feedback.

[15] Then the students see the survey. the border around the survey will be replaced with an HTML header and footer containing the branding specific to that OnFORME installation.

1.1.3 Create the Courses, Add the Students and Surveys to Database

This example explains the step-by-step process by which Paula could have used OnFORME to collect the Lent Term feedback from the students. This example assumes that she is collecting information on 5 modules, named Module1, Module2, …, Module5. Also, for the sake of simplification, it is assumed that there are three lecturers who teach in each module, namely Lecturer 1, Lecturer 2, and Lecturer 3 (where "Lecturer" is their first name, and the number is their last name). Likewise, it assumes that there are 9 TP students, Student 1, Student 2, … , Student 9.

Paula would perform the steps listed below to set up the system to collect feedback from the students. The step numbers correspond with the circled numbers in the listed figures, which represent where the action takes place.

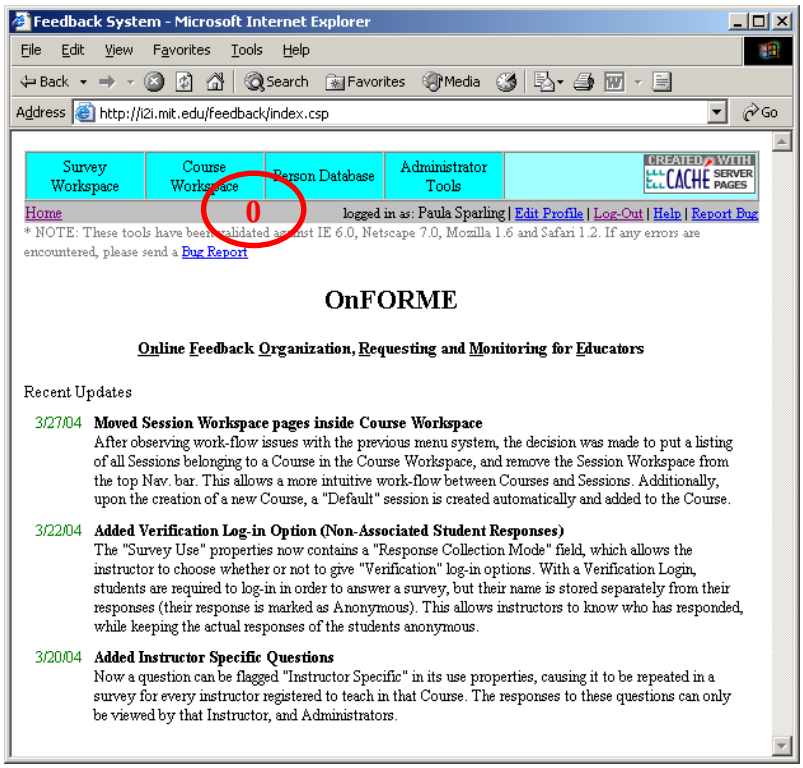


Figure 1-5: The OnFORME Home Page

0) After logging into OnFORME with her web browser, Paula would be brought to the OnFORME homepage (Figure 1-5). To begin creating the new modules in the database, she would click on "Course Workspace" in the top navigation bar.
1) She would then be brought into the Course Workspace, where she would click "Create New Course" (Figure 1-6).
2) Next, she would choose the Subject from which to create a new Course (Figure 1-6). In this case, her first selection would be "Module1"

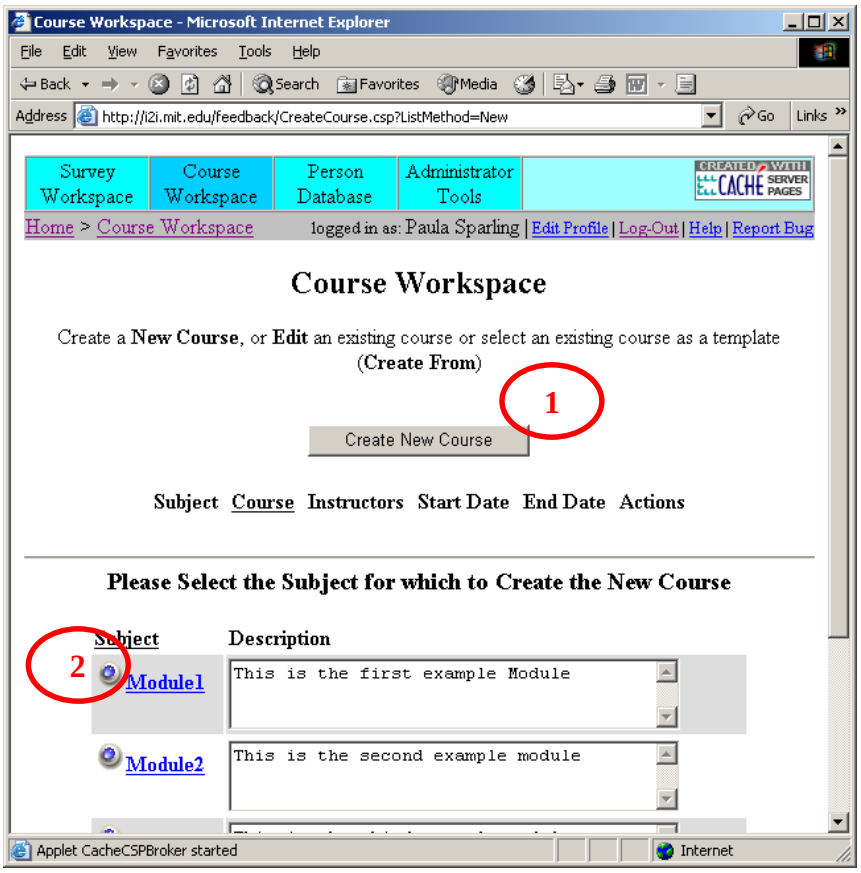


Figure 1-6: Creating a New Course

3) She would then be prompted to enter a name for the Course (Figure 1-7). Here, she would specify that it was Module1 given in the Lent Term of 2004.

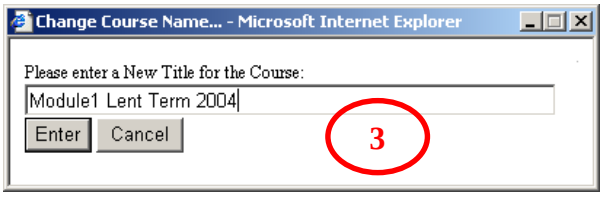


Figure 1-7: Naming the new Course

4) At this point she would be in the Course Workspace in OnFORME, where she could put in the details particular to this Module. First, she would put in the course start and end dates, as well as any other descriptive information that she chose to include (Figure 1-8).
5) Next, she would add the three lecturers to this course, by clicking on the "Add Instructor" button, which would give her a list of the people in the system. She would choose "Lecturer 1", "Lecturer 2" and "Lecturer 3" (Figure 1-8).
6) Similarly, she would add the appropriate students to this course, using the "Add Student" button (Figure 1-8).

7) She would then edit the default session in this course, by clicking on the "Default Session" link, or the "Edit" link in that row (Figure 1-8). The Session is the part of the course that holds the information pertaining to surveys used and email announcements. There can be multiple sessions within a course to allow for easier organization.

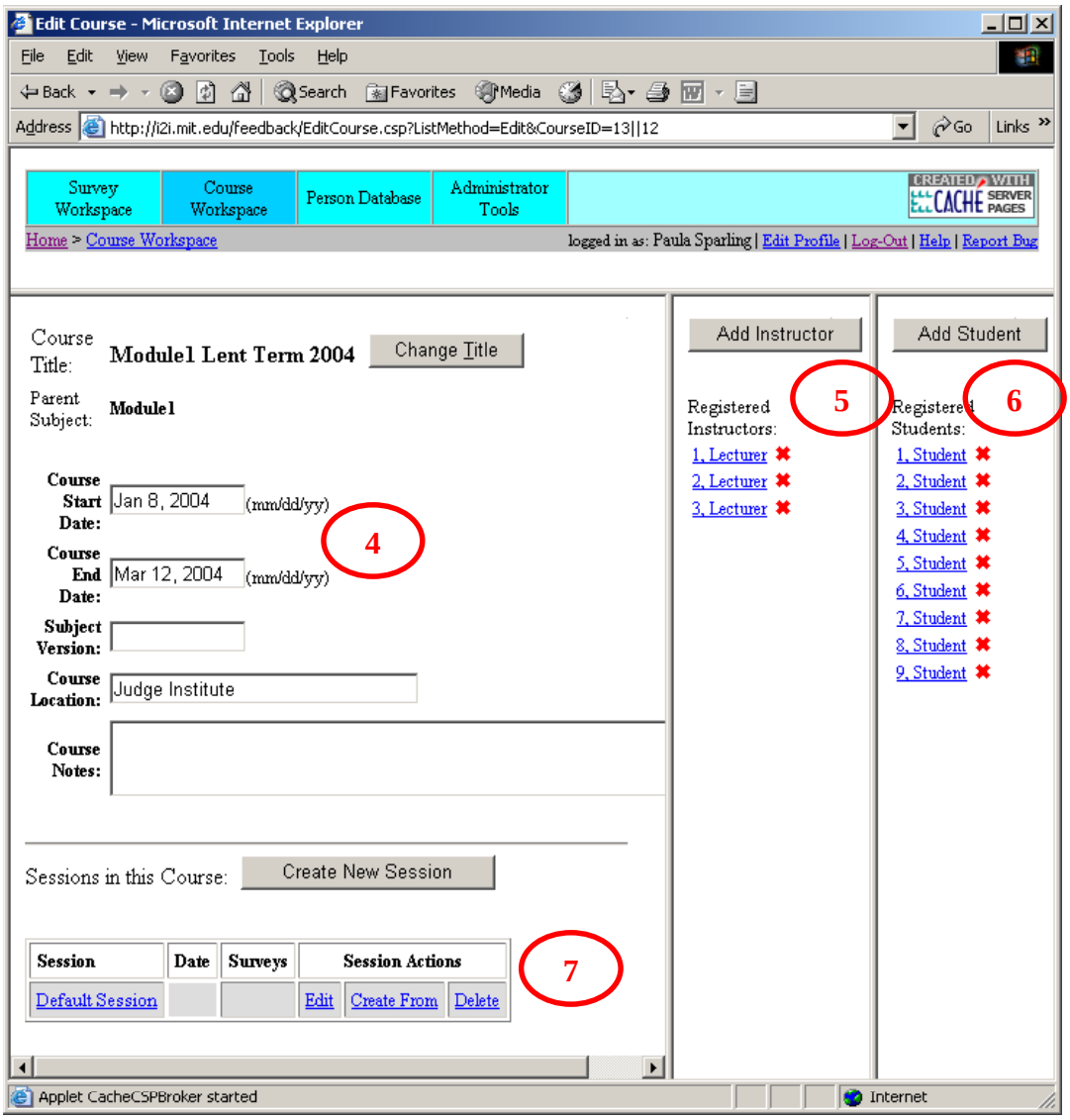


Figure 1-8: Setting up the Course Details and Course List

8) Clicking on the Session would bring Paula to the "Edit Session" page, which tracks surveys, emails, and class attendance. Paula would first add whatever descriptive information she chooses (Figure 1-9).
9) Next, she would click "Add Survey" to attach a survey to this session (and therefore to the module or course) (Figure 1-9). Clicking this button will bring her to a list of Surveys from which she can choose, and she would choose the MPhil Course Feedback Survey.

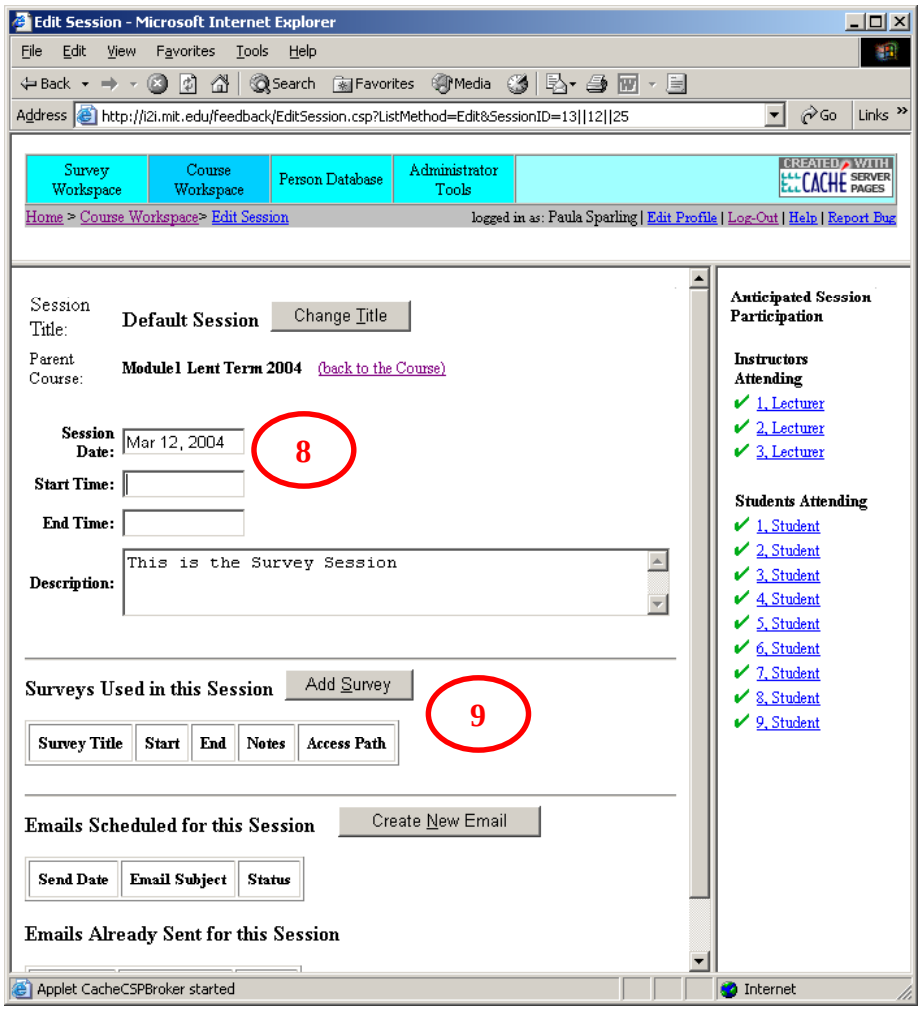


Figure 1-9: Setting up the Session Details and Adding a Survey

10) After selecting the survey she wishes to use, a page will pop up which allows her to configure the way in which the survey can be accessed (Figure 1-10). Here, she will specify that a "Verification Login" should be used, which will allow her to identify which students responded to the survey, while keeping their answers anonymous. She might also choose to specify start and end dates between which the survey is active. Having completed this, she will close the window.

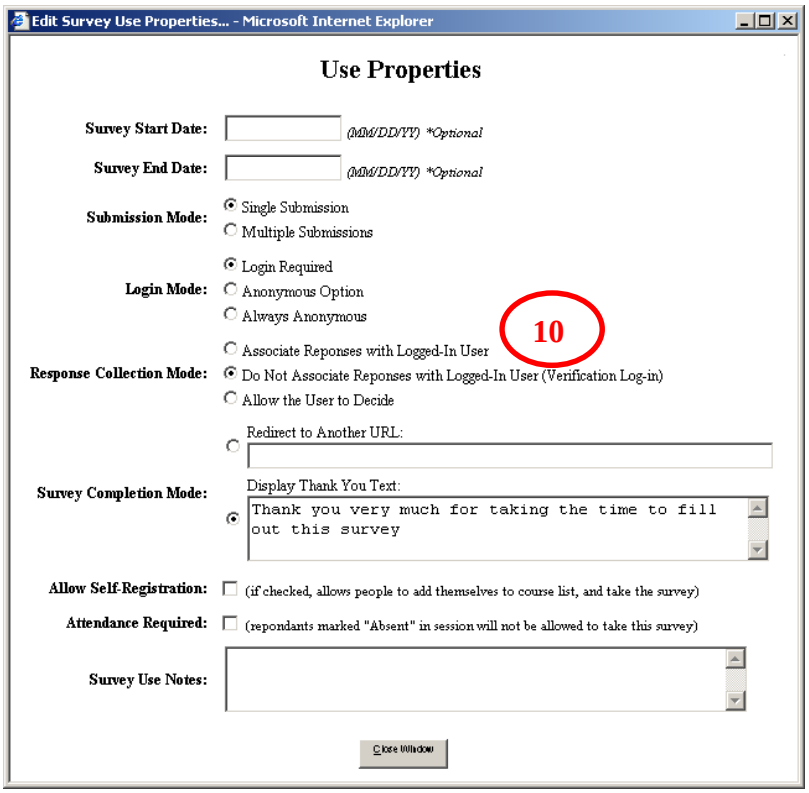


Figure 1-10: Specifying how the Survey is Used

After the window is closed, she will see that the Survey she has chosen has been added to the Session window, and that a unique URL "Access Path" has been created so that students can get direct access to the survey (Figure 1-11).

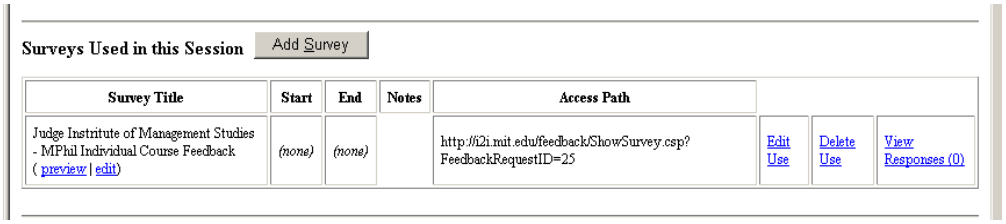


Figure 1-11: The Survey Listed in the Session

The process described above must be repeated for all five modules that the TP students took during that term. The preceding 10 steps are relatively simple and straightforward, and she should be able to easily input all five modules in less than 30 minutes. At that point, Paula has set up all of the pieces of the database, and she is ready to inform the students that the surveys are ready to be taken.

Prior to going to the next step, it is instructive to first stop and take a look at what has taken place in the process of attaching the survey to the course. Under the paper-based survey system (Figure 1-1), information on each of the three lecturers was collected in the same survey. This process is done dynamically in OnFORME, through the use of

"Instructor Specific Questions". The survey preview shown in Figure 1-4 shows the initial six questions are prefaced with the text "Referring to {Instructor Name Here}:" while the questions common to the whole course do not include that text. The initial six questions are flagged as Instructor Specific, which means they adapt to reflect the course in which the survey is used.

To illustrate this, a snapshot of the first couple of questions is shown when the survey is not attached to a course (Figure 1-12). This should be compared to what those same questions appear as when the survey is used in the Module1 course (Figure 1-13).

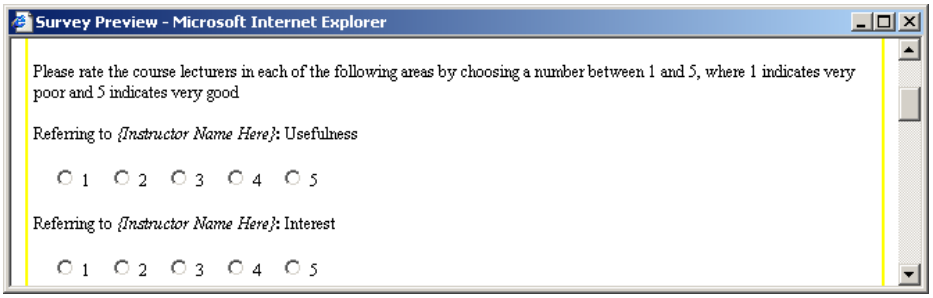


Figure 1-12: Instructor Specific Questions - Unattached Survey

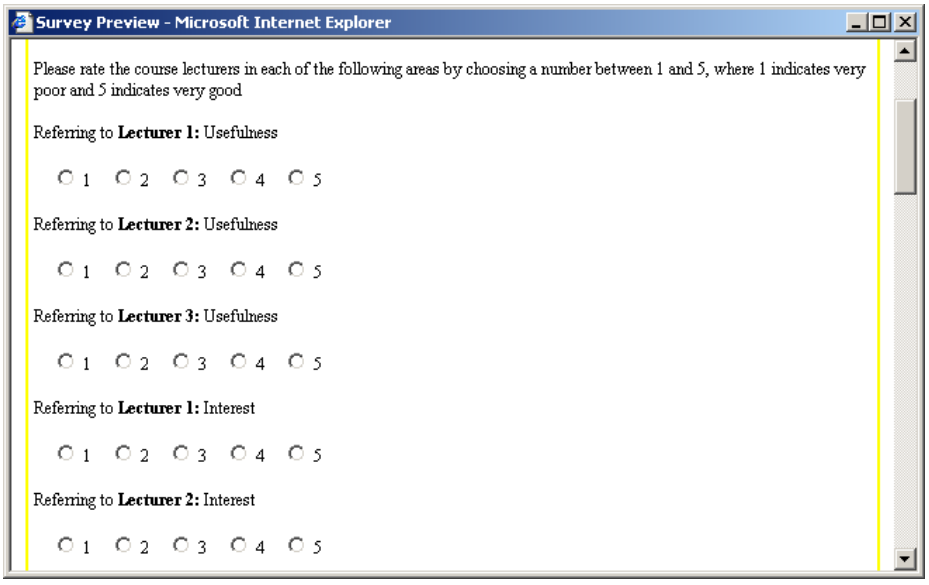


Figure 1-13: Instructor Specific Questions - Attached Survey

Note in the figure above, that the instructor specific questions have been rendered multiple times in the browser – once for every instructor registered for the course (refer to the list in Figure 1-8). Each time they are rendered, they include a direct reference to a different instructor by their first and last name (remember that in this example "Lecturer" is used as a first name, and a number for the last name). This means that the survey dynamically adapts to the settings in which it is used, which enables broad reusability (this is discussed at length later in this thesis).

1.1.4 Email the students with links to the Surveys

Once the system has been set up to receive the surveys, the next step is to make an announcement to the class asking them to respond. The steps Paula would take to do this are continued below.

11) In the Session window, Paula would click "Create New Email" in order to create an announcement that will be sent to the class (Figure 1-14).

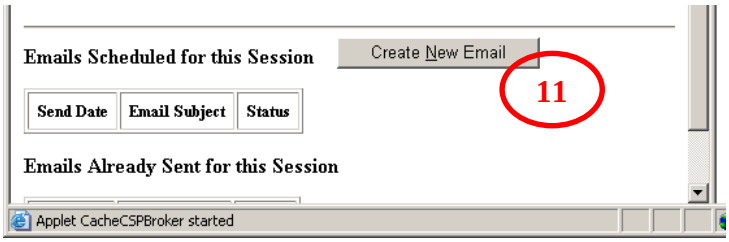


Figure 1-14: Creating an Email

12) The Session E-Mail window would open, in which Paula would type in an email subject and text body, as well as select a date on which the email should be sent (Figure 1-15). When she is finished, she would press the "Save & Close" button.

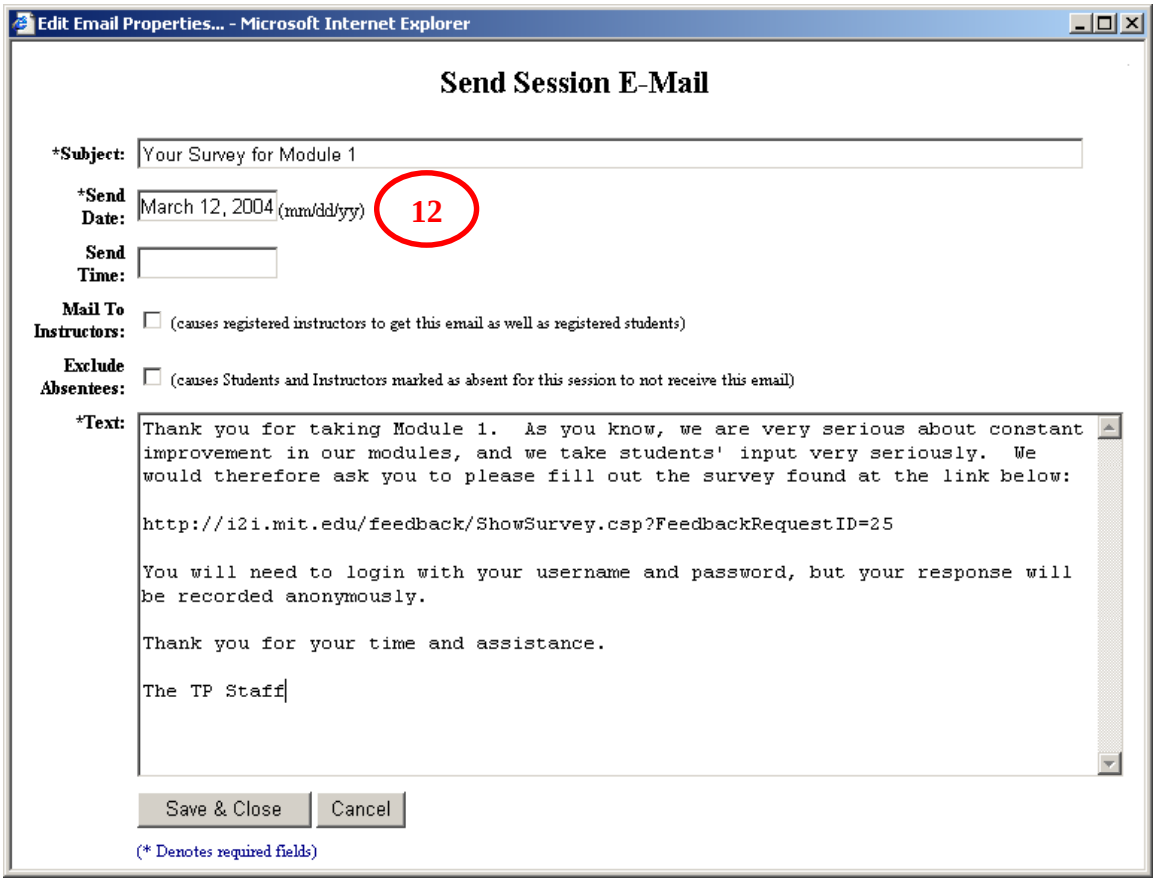


Figure 1-15: The Email GUI

OnFORME has the ability to queue email announcements and have them sent to the students in a course at a specified time and date. The email will be sent to all of the students in the class whose profiles contain a valid email address. Text of the message can be anything desired, and in this case Paula writes asking the students to fill out the survey, and she includes the unique survey URL that was generated to allow access to this use of the survey.

Besides this first announcement, Paula might also want to consider queuing a second announcement for two days later. This email would be a reminder sent to the entire class, asking that if they have not done so, they need to respond to the survey. After the emails are created, they are listed in the Session window with various control options (see Figure 1-16).

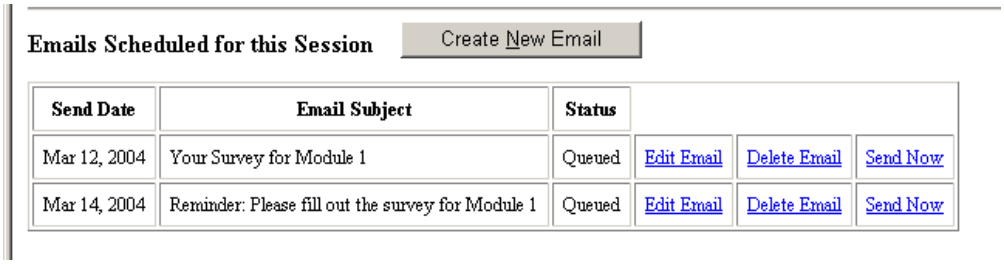


Figure 1-16: Emails Queued to be Sent

Since the process of queuing, addressing and individually sending emails is done automatically in OnFORME, it would take Paula only a couple of minutes to set up the announcements to go to those registered for the five modules

1.1.5 Email those a week later who have not completed the Survey

Suppose a week has passed, and Paula wants to single out those students who have not yet responded. She can do so with the following steps, and provided that not too many of the students are negligent in responding, the assumption is that she could take care of follow up in under 30 minutes.[16]

From the Session window, Paula can see how many students have responded to date, and she can click on the "View Responses" link to go to the response reporting pages (Figure 1-17). Alternatively, she can follow a similar link in the Course Workspace window.

[16] One of the pieces of intended future functionality in OnFORME is the ability to send queued emails on an individual basis to students based on whether or not they have responded to a particular survey. When this functionality is implemented, the manual follow-up requirements for Paula will be eliminated.

Figure 1-17: Navigating to the Survey Responses

13) The first Response page to which she will be brought is the "Response Summary" page. This page shows the executive summary of the results, and it will be discussed later. To find out who has not yet responded to the survey, she needs to select "Response List" under the Change View dropdown box (Figure 1-18).

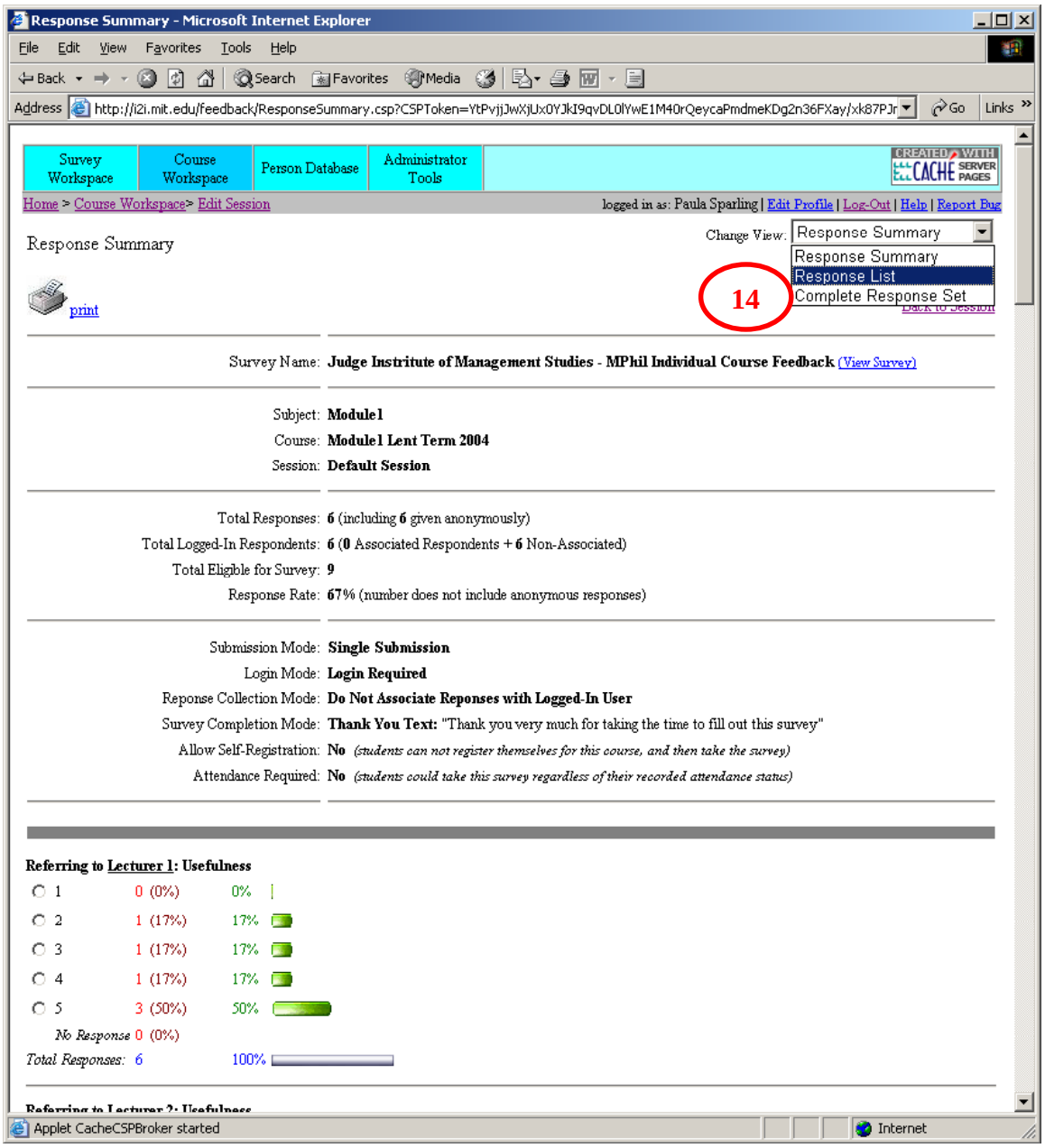


Figure 1-18: The Response Summary Page

14) The Response List page lists all of the responses collected for a survey use (called a "Feedback Request" in OnFORME terminology). Since Paula used a "Verification Login" for the students, all of the answers are shown as anonymous responses, but a separate list is kept detailing which students have and have not submitted a survey (Figure 1-19). Paula can click on the students' names that have not responded yet, and she will see their contact information. From this, she can either call or email them to remind them that they need to fill out the survey.

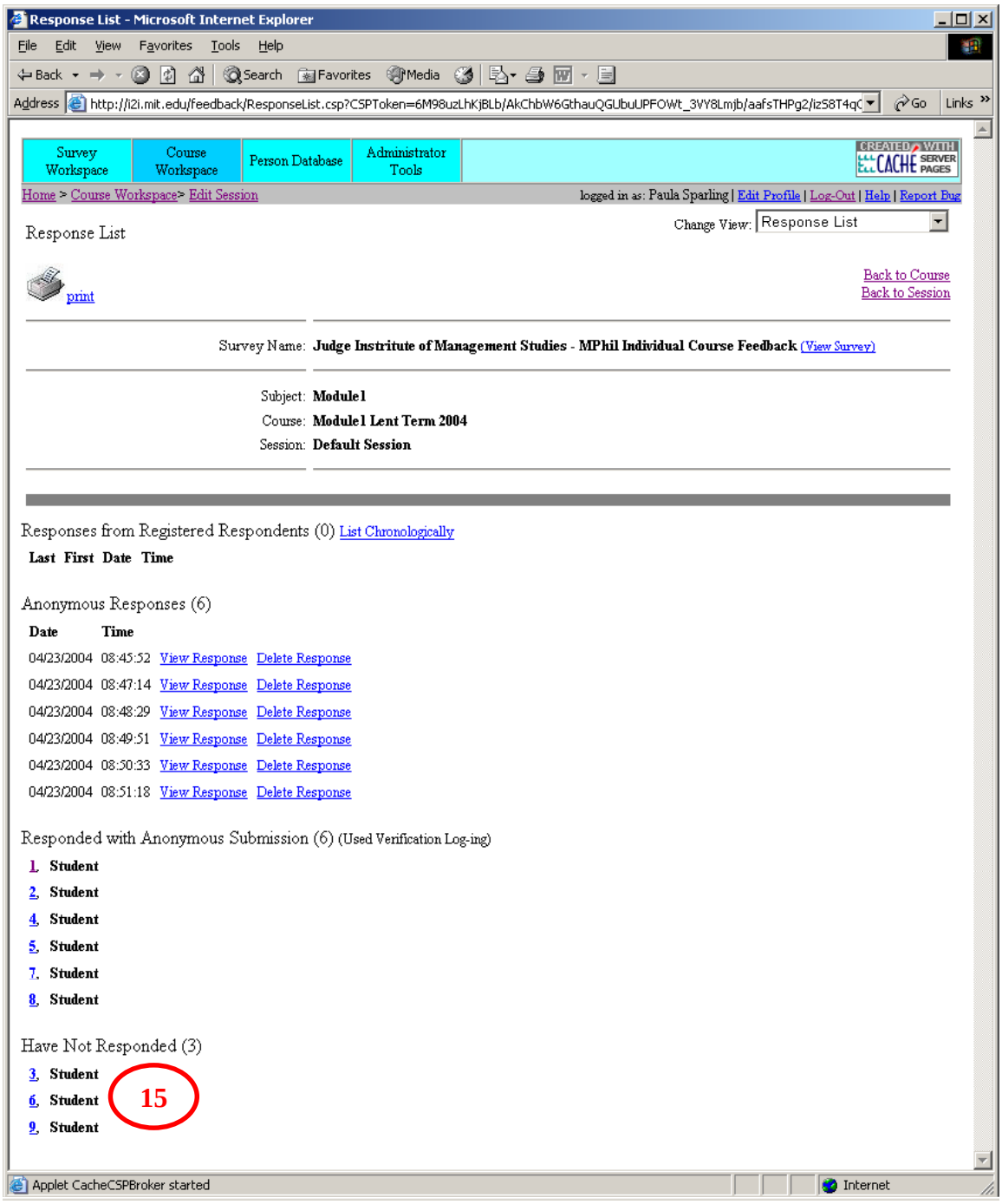


Figure 1-19: The Response List Page

To chase the non-respondents from all five modules is estimated to take about 30 minutes, since Paula can immediate know who has not responded, and have their contact information readily available along with that list.

1.1.6 Chase Lecturers to Get Comments

The next step in the feedback process for Paula is soliciting comments from the lecturers who care to respond to the comments and ratings made by the students. Previously, this had been an entirely manual process, where Paula made response reports for each of the lecturers (after hand-entering all of the students' surveys), and then she sent the reports to the lecturers, asking that they respond. She would then have to compile those responses.

With OnFORME, this process could be almost entirely automated, as the responses are totaled and correlated automatically, and a feedback portal can be created to collect the instructors' comments. Automating this process requires an additional survey to which the lecturers respond with comments. This survey has a single text box in which they can write their response to the feedback they had received.

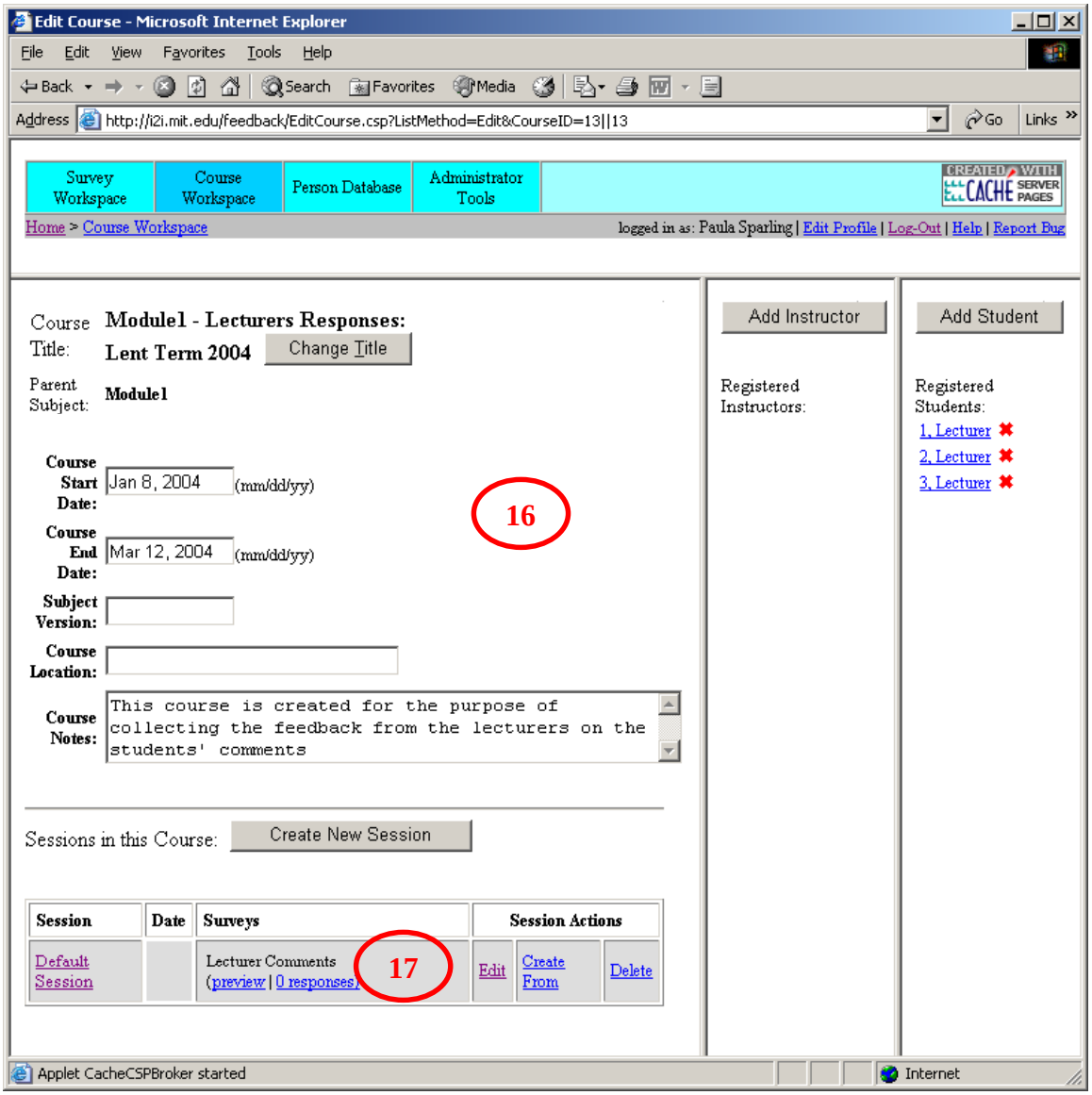


Figure 1-20: Setting up a Proxy Course for Lecturer Comments

15) Paula would have to create another Module1 course, which would act as a proxy used to hold the survey for comments. The reason that another course must be created is that survey access is limited to the students registered for a course (unless it is an anonymous survey). Therefore, creating a Module1 Comments course in which the three lecturers are listed as "Students" provides an easy way to control access to the survey and limit it to the lecturers while collecting their login information. The course would be created, the lecturers added as students, and the appropriate descriptive information would be entered, as shown in Figure 1-20.
16) Paula would then add the "Lecturer Comments" survey to the Default Session in the same manner discussed in the previous section (steps 9 and 10). The survey would be listed in the Course as shown in Figure 1-20.
17) When Paula uses this survey to get the lecturer comments, there are a couple of changes that she should make compared to the last survey used. Unlike with the students, it will be necessary to know which lecturer said what, so rather than setting up the survey with as a "Verification Login", she would select "Associate Responses with Logged-In User" (Figure 1-21). Additionally, she would put in a start date and an end date to control the period in which lecturers are permitted to comment.

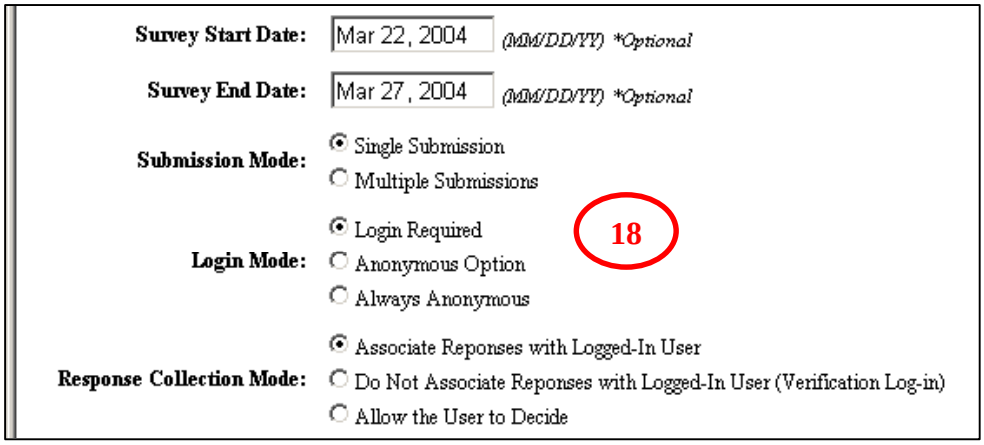


Figure 1-21: Specifying the Usage Parameters for Lecture Comments Survey

18) Paula would also need to queue an email announcement in the new Course to inform the professors where they could see the student responses, and where they need to fill in their comments on the evaluations. Like before, she would set up an email, an example of which is shown in Figure 1-22. If she so wishes, she could queue a reminder email to the instructors as well.

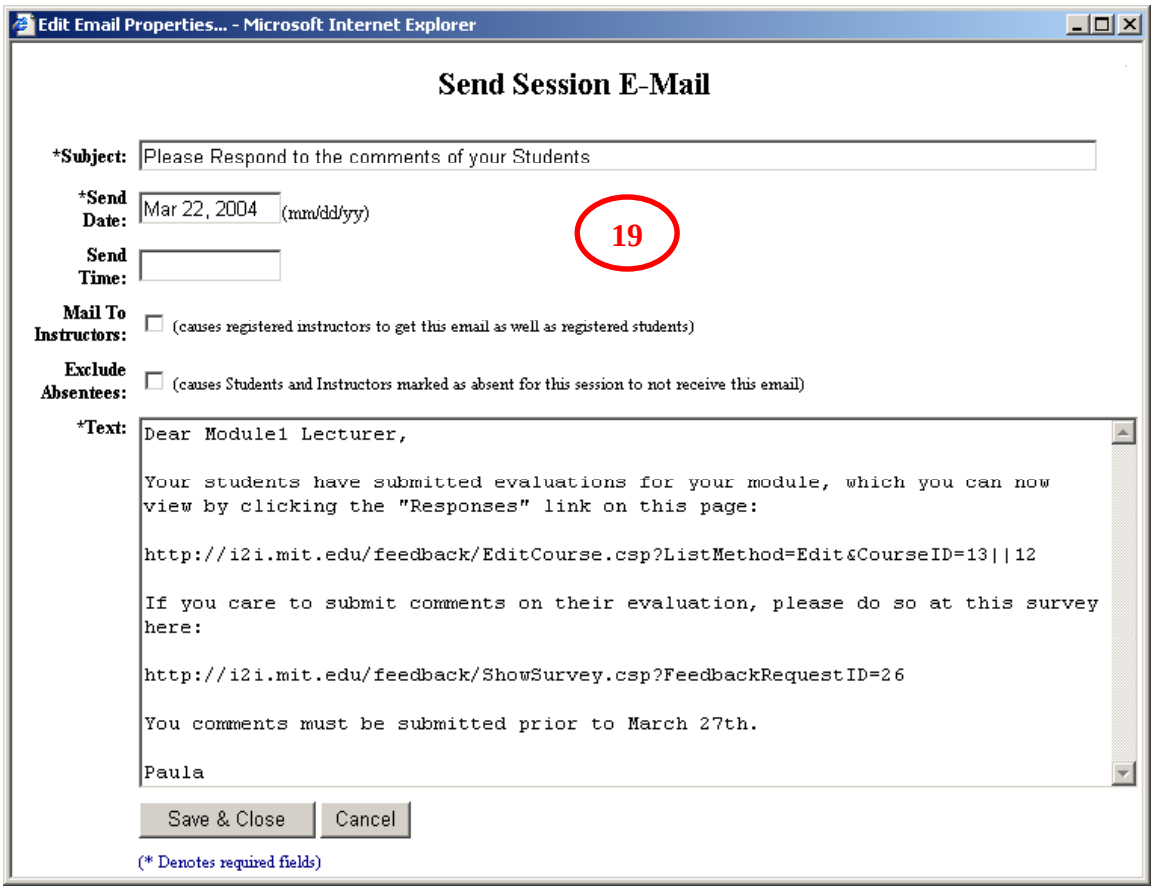


Figure 1-22: Email Message to Lecturers

After the instructors receive the communication, they will be able to go into the system and read the responses from the students. However, any responses that are tagged as Instructor Specific will only be made partially visible to the instructors. That is, each lecturer will only be shown the answers specific to themselves, and they will not see any answers given for their fellow lecturers. This protects the confidentiality of the responses, while giving the lecturers the ability to enter the system and view their responses (so that Paula doesn't have to photocopy all of the results for each lecturer, as was the case with the previous system). Alternatively, Paula is able to see all of the responses for all of the instructors due to her role as a system administrator. In this way, the interfaces enforce user-specific reporting of the survey results.

Soliciting comments from the lecturers of all five modules should not take longer than 30 minutes.

1.1.7 Print Final Report

Following the submission of the lecturers' comments, the last step that remains is to create a final report of the results and comments. To do this, Paula would first navigate to the response summary page (as was previously discussed). Then there is only one step left for Paula to complete the process.

19) From the Response Summary page, Paula could click on the Print Icon to print the executive summary for that use of the survey (see Figure 1-23). Alternatively, she could cut and paste the contents of the report into a word processor if she wished to change the formatting or add comments. She would then print out the Response Summary page to the Lecturers Comment survey (Figure 1-24), which could be appended to the end of the student evaluation report.

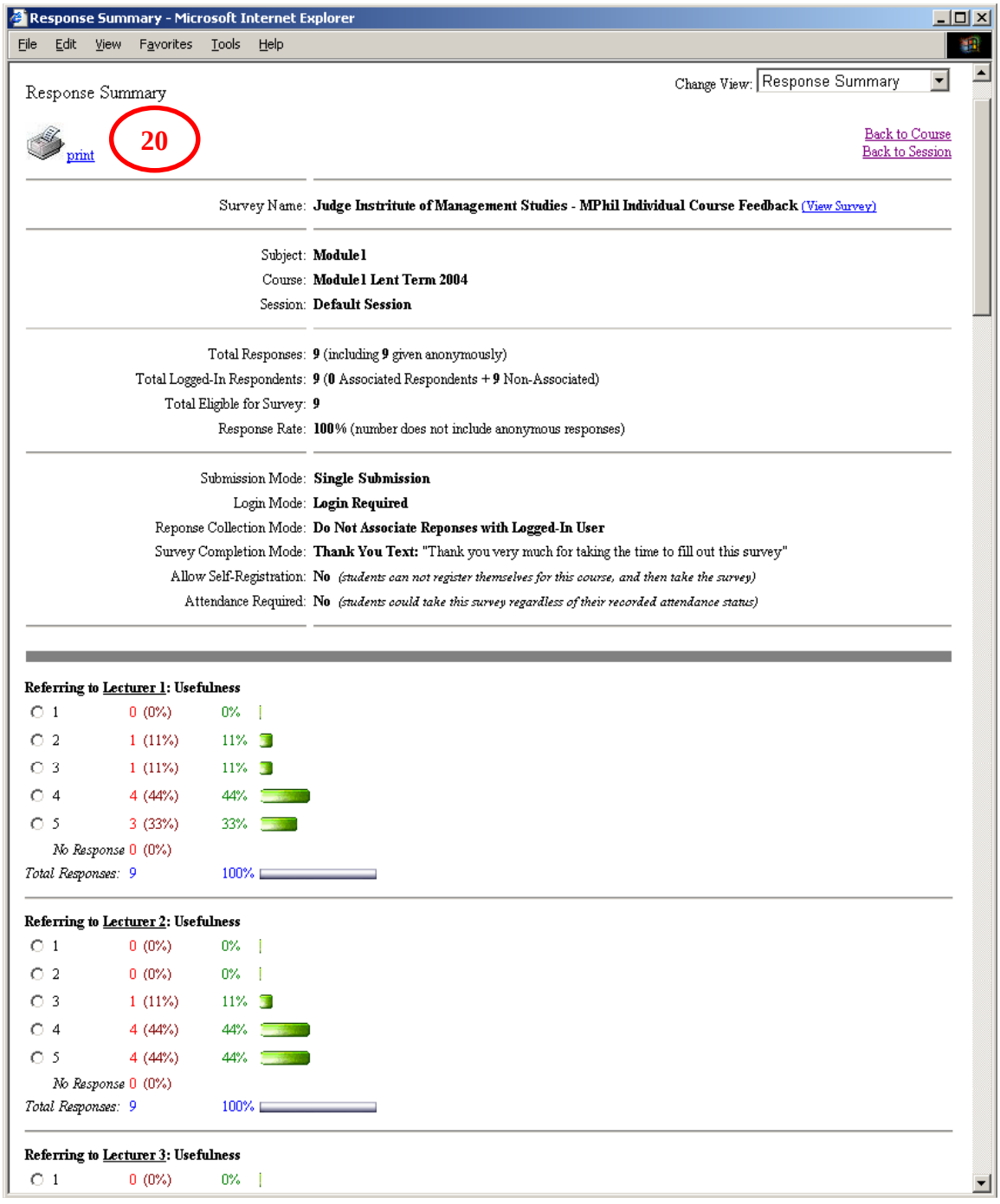


Figure 1-23: The Final Response Summary of the Students' Evaluations

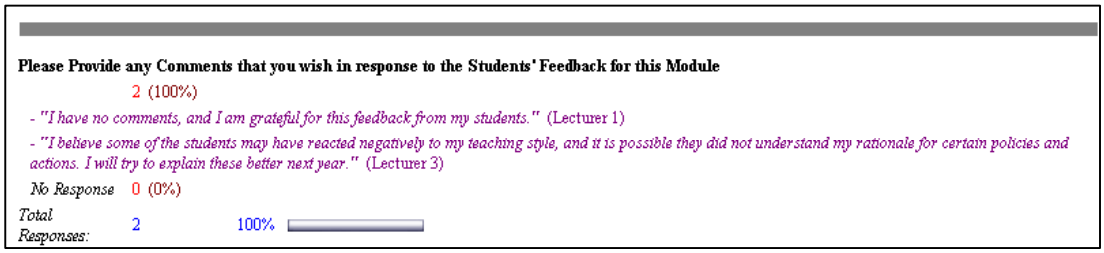


Figure 1-24: Summary of Lecturers' Comments[17]

As the survey results are pre-summarized, the report preparation time would be minimal for Paula, who could either print the summaries as they are, or could copy them into another document for further work. Either way, preparing the finished report should not take more than 30 minutes given the work that OnFORME does for her.

1.1.8 Conclusions

This example showed how Paula Sparling, the Course Administrator of the TP MPhil program at Cambridge University, could reduce the time she spends each term on student evaluations by 92% (22 hours). The example does not attempt to address the ways in which the content of the collected feedback might be made more useful, or other ways in which the feedback process could add more value in helping TP to improve its curriculum. Further changes to the survey content could provide additional value to the program above what the process changes can provide.

This is just one of many ways in which the OnFORME system can be used to collect information from students on learning processes and learning materials, which can be used by educators to improve their pedagogical efforts. Many other uses exist, some of which are discussed in this thesis, and others that have yet to be discovered.

1.2 Organization of Thesis

This thesis endeavors to communicate both the technical and social use findings associated with the creation of OnFORME. It is broken into four sections in an attempt to simplify the presentation of the learning.

The first section pertains to the system as a whole. Following this introductory chapter, a brief overview is given of related work in the fields tangential to the development of OnFORME. The following chapter gives the history of the OnFORME development and an overview of the initial organizations in which it has been used.

The second section of this thesis focuses on the technical findings gleaned from the application development. This begins with design overview, and the following two chapters discuss the intricacies of the underlying data model and how it was developed. Following this, a chapter then examines the user interfaces at a high level. Next is an

[17] Note – there are three lecturers, but only two committed responses, so the "2" in this figure represents the fact that 2 out of 2 responses contained an answer to this question.

overview of agile programming methodology and its bearing on the development of OnFORME. Lastly, the findings of an in-depth competitive analysis are presented to highlight ways in which the OnFORME system represents innovation in this application market.

The third section of the thesis focuses on the implementation trials and what was learned from them. The first chapter discusses in detail the various test sites for OnFORME and the data collected from those sites. The following two chapters discuss the usage findings relating to the application and use of OnFORME.

The concluding section attempts to tie together both the technical and social finding. First, there is a general conclusions chapter highlighting the discoveries in each sector of learning. Then there is a chapter devoted to key areas of future work on the system. This is followed by the list of references, and appendices.

2 Related Work

This thesis touches upon a number of different fields, and the following sections contain a brief discussion concerning points of research in those fields.

2.1 Educational Literature on Web-based assessment

The general area of web-based assessment is one that has received much attention in the last several years as the use of the web has gained prevalence. Liang and Creasy recently published a study focusing on the use of online assessments in online classroom settings, and how instructors and students benefit from the online tools (Liang and Creasy, 2004). Mertler put forth a very interesting work that compared response rates between two methods of surveying (paper and web formats), and then performed a follow-up survey with those in the Web group that did not respond (Mertler, 2003). An earlier paper by Solomon addresses data quality issues with web based surveys (due to coverage bias etc), and then discusses ways in which the response rates of online surveys could be increased (Solomon, 2001). Finally, Ravelli has done significant amounts of work on anonymous surveying for the improvement of pedagogical processes. He has both published on this topic and produced FAST (Free Assessment Summary Tool) to aid instructors in online survey publishing (Ravelli, 2000) (Ravelli, 2002). It should be noted that while there has been much written on the use of these tools, the development efforts are much more customer-driven and the researchers and developers do not have the resources to go back to the literature because they have to focus on the customer's use cases (Angelo, 1993). This thesis builds upon the previous work by exploring the technical innovation behind a wider range of system functionality than what is currently available in educational tools. Additionally, it takes a close look the adoption and use processes, and the factors that can prevent web-based assessment from being implemented.

2.2 IMS QTI

The IMS Global Learning Consortium Inc. is an organization that focuses on developing and heralding common, open technical specifications for learning technologies. In an acknowledgement that assessment is moving to the web, IMS created the "Question and Test Interoperability Specification" (or QTI). QTI is an XML standard that describes assessments and questions. The purpose of the standard is to allow different assessment applications and Course Management Systems (CMS) to be able to share assessment content (QTI Whitepaper, 2000). Currently, the QTI standard is in version 1.2.1, but plans are currently underway for finalization of QTI 2.0. The Future Work section of this thesis discusses the steps to be taken to make OnFORME QTI compliant.

2.3 Agile development

Due to the time and resource constraints placed on the OnFORME development efforts, strategies and approaches were used which deviated from the traditional procedures of structured software engineering. These approaches are termed "agile" in the literature, and over the last couple of years there has been a strong emergence of support for agile methodologies. Agile methods are summed up in the principles of the "Manifesto for

Agile Software Development" which puts more of the focus on the customer and functionality than on the documentation and process (Beck et al, 2001).

Of the many agile methodologies that exist, the most well know is called "Extreme Programming," or XP. XP was pioneered by Kent Beck, and contains a discrete number of practices aimed at making software development more flexible while maintaining the quality of the result (Fowler, 2003). While the OnFORME development process was not carried out with the intent to be "XP" in nature, the way in which development progressed ended up including many of the XP practices. Therefore, an in-depth analysis of XP practices and their equivalents in the development of OnFORME are presented later in this thesis.

2.4 Principles of Feedback in Educational Systems

The focus of this thesis is on the creation and early adoption of an online feedback system and does not focus on the contents of assessments. However, many of the principles in general educational assessment theory will certainly be appropriate for further use of OnFORME. One of the pivotal works on educational assessment was written by T. Angelo and K. Cross in 1988. "Classroom Assessment Techniques: a handbook for faculty" included many of they types of assessments that were used in the initial usage trials of OnFORME. They present the concept of a "minute paper", which is a 60 second paper that the students write at the end of class highlighting their most important learning of the day (C. Kerns, Personal Interview, 2004). An analogue to this "minute paper" approach was used in one of the OnFORME usage trials in which the instructor sent out a weekly survey asking the students to briefly outline their key learning from the two lectures of that week. This thesis adds to the decades of work that has been done on feedback in educational systems, by providing another technical platform on which the general principles and methods can be tested in a digital world.

2.5 Methods of Inductive Case Study Research

The research behind this thesis was not intended be for the purposes of deductive hypothesis testing. Rather, the goal was to employ an inductive analysis across a series of nine trial sites, for the purposes of crystallizing observations and generating hypotheses. This approach of inductive hypothesis generation has been well documented (Cutcher-Gershenfeld, May 12th, 2004). Central citations are Yin and Campbell on case study research (see Case Study Research: Design and Methods, Vol. 5). Additionally, with the adoption trials of OnFORME, the case investigation was participatory in nature, which is a methodology well discussed in social research literature (Foote Whyte, 1991).

3 Background

The OnFORME system represents a research support partnership between the Engineering Systems Learning Center at MIT, and the Learning Services Department of InterSystems Corporation. The partnership was conceived in June of 2003 as part of an InterSystems Corporation internship program. At the onset of that program, Jim Breen (the director of the Learning Services) and Dr. Joel Cutcher-Gershenfeld (the director of ESLC) realized that they shared a common interest in having a tool developed that would facilitate easy survey creation and automate feedback collection for their respective programs. The interest of Mr. Breen was pragmatic – he needed a flexible tool that could be used to determine the satisfaction of the customers being trained by his department, and give him the feedback information necessary to make improvements. The vision of Dr. Cutcher-Gershenfeld was more ambitious – his desire was for a tool that would make all manner of feedback in educational environments easy to create, use, and manage for instructors, thus pushing toward a paradigm of "ubiquitous feedback" to aid pedagogical improvement.

The common interest of the two parties in a flexible feedback tool launched the development effort behind OnFORME, which was funded by InterSystems Corp. during the summer (with continued support through the '03-'04 school year), and supported and funded by Dr. Cutcher-Gershenfeld during the '03-'04 school year. The goal was the production of a software platform that would be fully owned by each and would benefit each organization following the completion of the project. Throughout this thesis, Jim Breen, Dr. Joel Cutcher-Gershenfeld (and the future users in their respective organizations whom they both represented) are referred to as the project "customers."[18]

3.1 Design and Development Constraints

From the beginning, OnFORME was considered an ambitious project. The goal was to build an entire standalone system that would meet a myriad of user needs, while staying within a variety of design constraints. These constraints impacted the style and speed of the development methodology and necessitated many system-level strategic trade-offs. These constraints were time, labor, and capital.

3.1.1 Time

The project was conceived in June 2003, and in order to finish its coding and collect social usage data with the application (for the TPP part of this thesis), it was necessary that the following timeline be strictly enforced:

- Project begins in June 2003
- Beta Release[19] by October 2003

[18] Late into the development stage, Dr. Bill Nuttall, the director of the Technology Policy Program at Cambridge University, UK, also expressed interest in testing out the system. Due to his status as a research partner of Dr. Cutcher-Gershenfeld, his use cases were also taken into account and he was considered a "customer" for the purposes of this thesis.

[19] A Beta release is a version of the software which contains most of the functionality and is mostly stable, and is evaluated in a controlled test environment by early adoptors.

- Production Release by January 2004

This timeline meant that prior to the Beta release, there was a four-month window in which:

- the users had to be queried for usage cases
- the system needed to be designed
- beta-level functionality had to be coded, tested and debugged

Then between the Beta and Production releases, there was a four-month window in which:

- the beta sites needed to be analyzed for unforeseen usage requirements
- the discovered usability inhibitors needed to be reworked
- all additional production release functionality had to be coded, tested and debugged

This was especially ambitious, considering the size of the team that would be working on this project.

3.1.2 Labor

From the beginning of the project, it was known that I would be the only person working on the OnFORME application. This meant that I would be fulfilling all of the following roles during the eight-month release schedule:

- project management
- system modeling
- application design
- interface construction
- application coding
- system testing and debugging

I would have at my disposal all of the technical expertise of the training and support departments at InterSystems, to help me climb the learning curve for my chosen implementation technologies and debug problems. Other than that I was on my own.

The development cycle was short to begin with, but the time restriction was aggravated because of my inability to devote all of my time to this project due to other commitments. This was not a problem during the four-month schedule leading to the beta - during that time I could devote all of my energies to the creation of OnFORME. However, during the second four-month development cycle, my time was split between the application work and my course load of thirty-six units. This relegated much of the work to being completed during school breaks.

3.1.3 Capital

Other than covering the cost of my research efforts, there was no budget for additional expenditures for this project. This meant that purchasing licenses for other products was not a possibility (thus simplifying the "buy vs. build" decision). Also, I would have to work with hardware already owned by the customers, and I could not expect any additional help to be hired to aid in the development process.

These three constraints were taken into consideration at every point during the design process. It was recognized that it would not be possible to cover all of the usage cases in such a short development cycle, given the size of the development team. However, the goal was to provide the functionality demanded by as many of the use cases as possible during the development time allotted.

3.2 The Social System Application Context

An important aspect of the OnFORME system is that it was not created in a vacuum apart from the reality of its intended contexts. On the contrary, it was designed and implemented with almost daily interactions between the development team (myself), and the social contexts in which it would be used. Various design decisions were tested with users as development proceeded, and immediate feedback of customer preferences was taken advantage of.

When constructing a software application, it is important to understand the social context in which it will be used, so that the development efforts may shape the product to conform to those contexts where appropriate. There are in fact a wide range of feedback collection tools which exist on the market, however the social context for most lie outside of the contexts of educational environments. Most are aimed at market research, customer satisfaction surveys, or general public polling. This means that the functional software structures supporting these applications from the periphery (e.g. the way assessment are organized, the way respondents are tagged and tracked, etc), are calibrated towards those social settings. In creating a tool to enable educators to improve their pedagogical processes and materials, it is a great benefit to have a feedback collection application that has been developed and tuned for use in educational environments. Therefore, it is important to understand the needs of the educational environments (both academic and corporate) that the OnFORME application was intended to fulfill.

3.2.1 Academic Educational Environments

There were a variety of uses that were envisioned for OnFORME within the university setting. Some of the ways in which the system might possibly be used are described below to illustrate the range of applications.

- Professors may wish to send weekly short surveys to their students in order to capture the learning highlights from the lectures for each week.
- Instructors may wish to understand the effectiveness of their teaching for certain "learning objectives". To capture this information they might send out a "pre-lecture knowledge assessment" prior to a lecture (to test the content understanding of the

class), and then do a "post-lecture knowledge assessment." The results of the two surveys would be compared in search of patterns of improved comprehension.

- An instructor may wish to gain a real-time snapshot of content understanding while giving a lecture, and would therefore direct his class to a single-question anonymous survey which reports in real-time the answer distribution of the class, thus allowing him to adjust the pace of the lecture.
- An instructor may wish to collect feedback on his teaching efficacy, and therefore send a survey to the class inquiring as to their opinions of his lecture clarity, quality of handouts, etc.
- An instructor may wish to do a learning exercise in which students working in groups assess each other on various group-skills, thus providing a learning opportunity for each member of the group as they submit themselves to the honest scrutiny of their peers.
- A program administrator may wish to discover the appropriateness of a required course event, and therefore send out a survey to the entire program querying opinions relating to the event in question.
- The author of course materials may wish to understand ways in which her materials are being used around an institution (or around the world), and therefore will send a survey to those people whom she knows are using the materials, inquiring as to their usefulness and requesting suggestions for improvements.
- An instructor wishes to open an "honesty portal", which is a short anonymous survey that students can use anytime during a semester to communicate grievances that they may have with the class.
- A course administrator is required by the department to conduct end of the semester assessments, and would prefer to do them electronically rather than via paper and pen, which would later necessitate hand keying the data.

This is just a sample of the ways in which feedback can be used in academic environments. Many of these situations were (or are planned to be) test trials of the OnFORME platform, at either MIT or Cambridge University (which became engaged as a test site using MIT's server towards the end of the OnFORME production cycle).

3.2.2 Corporate Educational Environments

Typically, when one hears the term "educational environment", the picture brought to mind is of full-time degree based educational institutions. However, this is not the only market for learning, as there is a rather large "corporate training" market. Producers of technical products give much of this training while educating (or certifying) customers on how to use a certain technology.

Technical training of customers is the mission of the Learning Services Department at InterSystems Corporation. Their team of technical trainers provides classroom based training, eLearning broadcasts, online tutorials, and technical certification services to customers wishing to learn how to use InterSystems software. They are educating 24 people a week (not including their Caché Campus, Tutorials, or Caché Entrée programs). As the corporate customer for the OnFORME system, the ways in which the Learning

Services department hoped to use the application played an important role in the development prioritization. Possible uses for the application are described below.

- If the director of training wished to know the perception of his technical trainers, he could require that each training session conclude with the customers being given a survey, in which they rate the trainer on certain important characteristics.
- If an eLearning instructor wished to check if their students were paying attention to the lecture, then they could do a short survey to spot-check the comprehension level of the students on topics recently covered in the discussion.
- If the author of an online tutorial wished to know how effective the tutorial is in teaching the central points, then she could embed within the tutorial web-redirects to a survey that asks basic questions on key concepts, and then displays the correct answer if the student answers it incorrectly (thus providing feedback to the author, and to the tutorial user).
- If the long-term value of a training class is to be assessed, then an instructor may wish to prepare a survey which automatically is sent to all students 6 months after they finish a class, inquiring as to their thoughts on the course after they have had a chance to attempt to apply their learning in production environments.
- If there is more than one instructor teaching a class, then there may be a need to give a survey to the students that inquires as to the quality of each instructor, but the responses would be stored in such a way that each instructor could only view her own feedback, while their manager could view the feedback for all instructors.

The corporate customer envisioned these uses and others, and an understanding of the structures and motivations of corporate training departments was instrumental in shaping a tool that could meet their needs. The use cases from both the academic and corporate training settings influenced the development priorities of OnFORME, while the greatest weight was given to those use cases which applied to both settings.

Part II – Technical Systems Components

4 Design Overview

This chapter discusses the high level architecture and the technology used in the OnFORME system. The chapter concludes with an overview of the software components in the completed system.

4.1 OnFORME Architecture Model

Due to the brevity of the development timeline and the heterogeneous environments of the initial customers, OnFORME was designed from the onset to be a vertical application (i.e. it would contain all of its own functionality, rather than rely on being integrated with another system, such as the registrar's system, etc).[20]

Consistent with the name of the application (Online Feedback Organization, Requesting and Monitoring for Educators), the finished OnFORME system needed to not only be able to create surveys, but also to contain the means to communicate requests for feedback to the members of a course, to organize the surveys within a structure consistent with the educational environment, to view the results in a helpful way, etc. In any application that is based on the collection, organization, and interaction of data, it is the data model that proves to be one of the most critical parts of the application design. The application is attempting to create a digital version (or "model") of some aspect of a non-digital system. In this case, it is modeling how feedback is collected in educational environments. Therefore, in order for the functionality of the finished application to parallel the behavior and functionality of the original system, it is necessary to deconstruct the social system into its core components. These components are used as a basis for the data model on which the software application is built.

There was a great deal of time and effort put into the model that comprises the data back-end for the OnFORME system. From the beginning, one of the central design goals was to make the system as flexible and intuitive as possible (as usability was viewed as a system necessity), and a model that diverges from the system it is meant to represent can be neither flexible nor intuitive. Therefore, the modeling efforts aimed at perfecting the data back-end will be a major focus of this thesis. The final model can be viewed as the composite of different "blocks", in which each block comprises one or more data objects (realized as classes in the system design). The block-diagram for the OnFORME architecture is shown in Figure 4-1.

[20] From a deployment standpoint, this has the advantage of being able to function autonomously, rather than to rely on other applications. However, there is also the substantial disadvantage of its having to duplicate functions that may already exist in the coursework support environment. Future work on OnFORME should include adaptors that allow it to exchange data with other systems, possibly through web services.

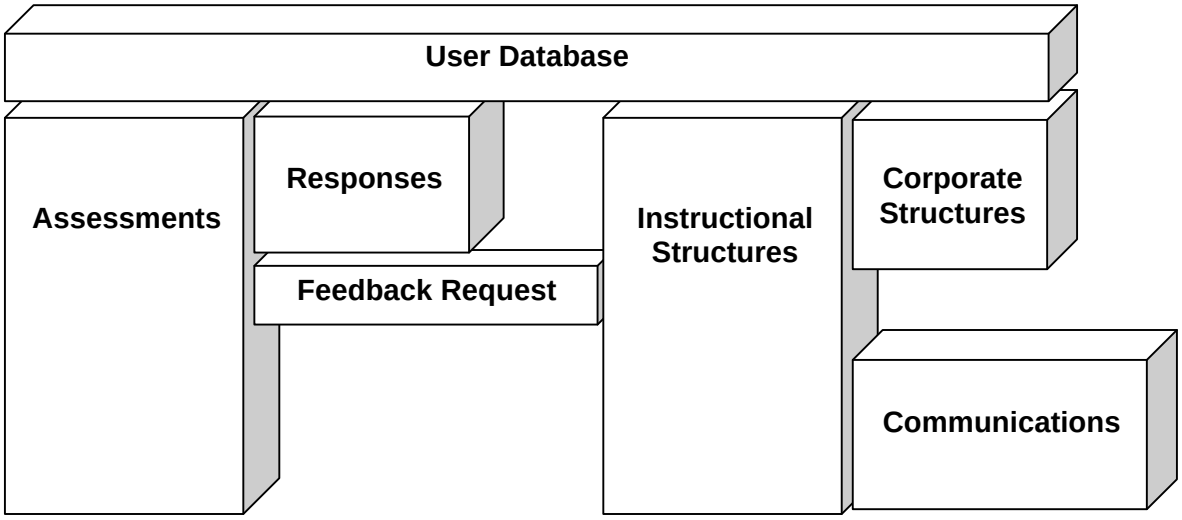


Figure 4-1: OnFORME Block-Level Diagram

Each of the blocks in Figure 4-1 represents a number of object classes, which contain the methods (functionality) and properties (data information) that make up the data model for that piece of the system. Each block interacts with a number of other blocks in a manner consistent with the modeled social system. When two blocks are adjacent to each other (in Figure 4-1), this means that the underlying objects communicate directly with each other across that boundary.

The User Database block contains the object classes that allow participants in the educational system to be "deconstructed" and modeled in a data-driven way. The objects in this part of the architecture represent the characteristics of people that interact with the system (e.g. First Name), their roles and responsibilities (e.g. student), and how they interact with the educational environments (e.g. class registrations). This block interfaces with most other parts of the data model, as the users create assessments, author responses, take classes, and are billed (for corporate training).

The Assessments block contains all of the objects that make up assessments. Throughout this thesis, the terms "Assessment," "Survey" and "Feedback Object" will be used interchangeably to describe the same concept – a web based form created for the purpose of asking questions of one party (usually a student), and delivering the answers to another party (usually an instructor). These questions could have "right" answers (as are contained in a quiz), could inquire about content understanding, could ask about opinions, or could request information on any number of other subjects. The central idea is that an assessment as a group of questions whose responses may be used to "assess" something (e.g. student comprehension, quality of learning materials, helpfulness of instructor's lecturing style, etc).

The Instructional Structures block contains the data elements and object definitions that are necessary for modeling the structures in which instruction takes place. These structures include information about materials taught, times of instruction, topics, etc. The Instructional Structures block interacts with the users in the database, as well as

the parts of the system responsible for communications and tracking corporate training data needs.

The Feedback Request block contains a single object class (`FeedbackRequest`) that ties together the major pieces of the architecture. This block ties together the Instructional Structures data with the Assessments and their Responses.

The Responses block contains the object classes that comprise a completed response to an assessment. These classes contain the data that is used to report results out of the application. Responses tie in with the Assessments, as well as the User Database and the Feedback Request block.

The Communications block contains the objects used for mass communication of a request for feedback. The objects in this class interact with the Instructional Structures block in order to obtain the information relevant to the "who" and "how" of communication.

The Corporate Structures block contains the object classes that are included in the architecture to model the additional data and processes relevant to the Learning Services Division of InterSystems. These classes include the capability for tracking the logistical details of customer-site training, maintaining a record of the companies for which students work, and monitoring the status of eLearning subscriptions purchased by students. These classes may be relevant at other universities that employ a fee for course structure (e.g. Executive Training). The Corporate Structures block is connected to the User Database and the Instructional Structures.

4.2 Selected Implementation Technology

In order to achieve the project vision and timeline, there were several high-level goals that had to drive the design and implementation decisions. These goals were:

- Rapid architecture construction
- Rapid application development
- Easy integration between the database back-end and the web front-end
- Low cost components
- Always keep the customer's needs as top priority

A major focus of this design effort was responsiveness to customer needs, which heavily influenced the implementation decisions as well as the final feature sets. The selection of the technology on which to implement the OnFORME system was heavily influenced by customer preference.

InterSystems Corporation (one of the project sponsors) is a mid-sized software company whose primary product is an object-oriented database and rapid application development environment called Caché. Since InterSystems' desire was to use OnFORME as a support base for internal operations, their strong preference was that the application be built on their Caché platform, as it would achieve the following goals for them:

- Allow for easier integration with other internal applications (also built on Caché)
- Allow for easier maintainability owing to the fact that anyone technical in the company could service the application, since they were familiar with the underlying technology
- Prevent their having to purchase software licenses for another database or application development suite

From a programming perspective, I had extensive prior experience with object-oriented design (specifically within the Microsoft paradigm), but I had no real experience with databases or web technologies. Therefore, the majority of the technologies that would be used in application development would have to be learned from the ground up. This placed a high importance on the speed at which I could learn a technology. Therefore, since I would be working at InterSystems during the first three months of the development timeline, using Caché as the development platform gave the additional advantage of my being able to use the InterSystems learning materials and in-house expertise.

Dr. Cutcher-Gershenfeld at MIT did not communicate a platform preference for OnFORME. The design goals and the development schedule where emphasized as priorities, with special attention given to the need for the production system to not include expensive licenses. This goal would be attainable were Caché used, as InterSystems offers free unlimited licenses to educational institutions using Caché for teaching or research. This arrangement is part of the Caché Campus program,[21] which Dr. Joel Cutcher-Gershenfeld was willing to join.

Due to my need to learn new development technologies, the strong preference of InterSystems, the advantages of the Caché platform (discussed in the following sections), and the fact that using Caché does not interfere with any of the other development goals, the decision was made early in the process to develop OnFORME in Caché.

4.2.1 Description of Caché

Caché is the "post-relational" database created by InterSystems Corporation. InterSystems has been in the database market for 25 years, and their database currently runs behind approximately 60% of the world's health care information systems. Caché is the latest generation of the MUMPS (or M) programming language that was developed at Mass General Hospital several decades ago. Caché focuses on maximum scalability and reliability, while not sacrificing database access speed (one of their requirements is that each release of Caché be faster than the last).

The Caché database engine allows Caché to be both an object-oriented database and a relational database at the same time. The native programming language is called "Caché Object Script" and the product also supports a variant of Basic language called "Caché

[21] For more information on the Caché Campus Program, see http://www.intersystems.com/cache/education/cachecampus/index.html (correct as of 4/14/2004)

Basic". The various strengths of the platform that aided in the creation of OnFORME include:

- Full-features Integrated Development Environment (IDE) similar to Microsoft Developers Studio
- Integration with Microsoft Visual Modeler for ease of object-oriented development modeling.
- Object-binding support for most of the industry-standard object technologies on the market (see Figure 4-2), including XML for easy data export.
- Integrated web gateway, which allows web development with direct connections to objects within the database (Caché Server Pages).
- A large number of built-in libraries supporting web, email, web services, and scheduling functionality.
- Very robust database storage, which scales easily to hundreds of thousands of concurrent users without changing application code.
- An extremely fast database engine, capable of supporting hundreds of thousands of accesses per second.

Besides being a technology that supported all of the necessary components of OnFORME in an integrated way, Caché allows for future extensibility of the system (through web services, ODBC, .NET bindings, etc), which will create a broad range of options for future interoperability and functionality demands.

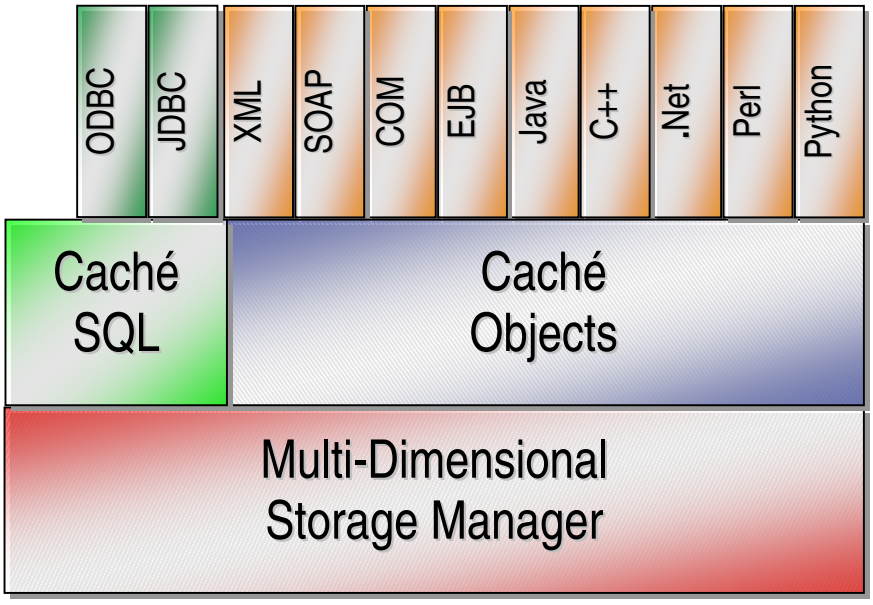


Figure 4-2: Bindings for Caché Data Access for Caché 5.1

4.3 Software Components

The OnFORME system consists of:

- 44 Caché Object Classes containing
  - Caché Object Script methods
  - SQL Queries
  - JavaScript Functions
- 74 Caché Server Pages[22] (CSP) containing
  - HTML
  - DHTML
  - JavaScript and JavaScript Functions
  - Caché Object Script
- Over 16,500 lines of code

All of the classes are contained in the `Feedback` package. All of the CSP pages are included in the `/Feedback/` application space, and the Help system pages are contained in `/Feedback/Help/`. All of the server-side methods (those which are not encapsulated in a class) that are referenced by the CSP pages are contained in the `CSPUtils` class, which is the super-class of all CSP pages in the application.[23]

4.4 Future Maintainability and Open Source Distribution

As OnFORME is the product of student research, many issues arise surrounding the sustainability of the software following the graduation of its author. Any software project can experience challenges for future adaptations if the sole architect and designer has moved on to other things. However this problem is compounded in academic software development since often the only designer with a vested interest in the work is the original student developer.

In some ways, OnFORME stands a greater chance of continued use than most student projects, due to its being created specifically for two customers that have a vested interest in its functionality. As they already have the system installed and are actively using it, it will likely remain in use for some time, provided it continues to yield value to them. One of these sites (InterSystems) is virtually assured a long-term reliance on this application since the developer (myself) will be working at that company, and can be called upon to make adjustments to the code or fix bugs. Additionally, a third customer is in the process of setting up a server – acting as an additional location in which the application can remain in use.

However, the desire is to see OnFORME achieve wider user, and continue to be improved in its functionality and feature sets. And, as neither the developer nor the

[22] CSP is the analogous to Microsoft's ASP technology, which enables server-side preprocessing of web pages.

[23] Early in the design process, it was logical to encapsulate all of the server-side methods in one utility class, providing a single point of reference for debugging and creating new methods. However, the number of methods and the number of CSP pages has increased to the point that there are now downsides to having to recompile all pages anytime a single function is modified. With future development, it might be wise to refactor this code and push any methods used by only one CSP page, out of the utility class and into that page.

initial customers (Dr. Cutcher-Gershenfeld and Jim Breen) wished to be in the application support business, the only way that this desire could be achieved was by making the source of the application freely available to anyone who wishes to adopt it for use. Therefore, it was decided to release OnFORME as open source, and attach a license to it that would allow for the broadest possible use of the code.

OnFORME has been approved for hosting by the open source online clearinghouse Source Forge (www.SourceForge.net). There, the code has been published under the "MIT License". The MIT License allows for use, modification, sale, and publication of the OnFORME code, provided that any altered code is published with the original OnFORME copyright statement and license agreement. Source Forge also provides web hosting for a project homepage (onforme.sourceforge.net), bug tracking services, messaging boards, and community communication[24].

The other advantage of the open source approach is that it allows for greater customization. Due to the agile development processes used to make OnFORME, it was not feasible to make the system broadly customizable. Therefore, adding new database fields, or changing the layout of the pages will require a change of the code and a recompile of the system. Without the source code, new installations would be forced to either accept the rigidity of the system, or request specialized changes (which would not be possible without a team supporting the code base). However, since the new users will have a copy of the actual code, they will be able to make the changes themselves. In some cases, customization will only require knowledge of HTML, while others changes may mandate that CSP be learned.[25]

The design and architecture information included in this thesis, along with the documentation contained in the code are intended to provide a foundation of background information. This information will be necessary for future developers that will maintain, adapt, and improve the OnFORME code base, be they at MIT, or at new installation locations.

[24] For more information, visit http://www.sourceforge.net/projects/onforme

[25] OnFORME was never intended to be an "out of the box" software package, as it is based on the client / server model and will require someone with technical ability to properly configure the server. Therefore, it is safe to assume that new installations will require a technical person being involved, who could then act as a resource for customizing the code and recompiling the application.

5 Modeling Educational Environments

The use of OnFORME as a vertical application in educational environments assumes that the application contains the necessary data architecture to model those environments. The architecture blocks relating to educational environments are:

- Instructional Structures
- User Database
- Communications
- Corporate Structures

These blocks (which are highlighted in Figure 5-1) are addressed in this order in the subsections of this chapter.

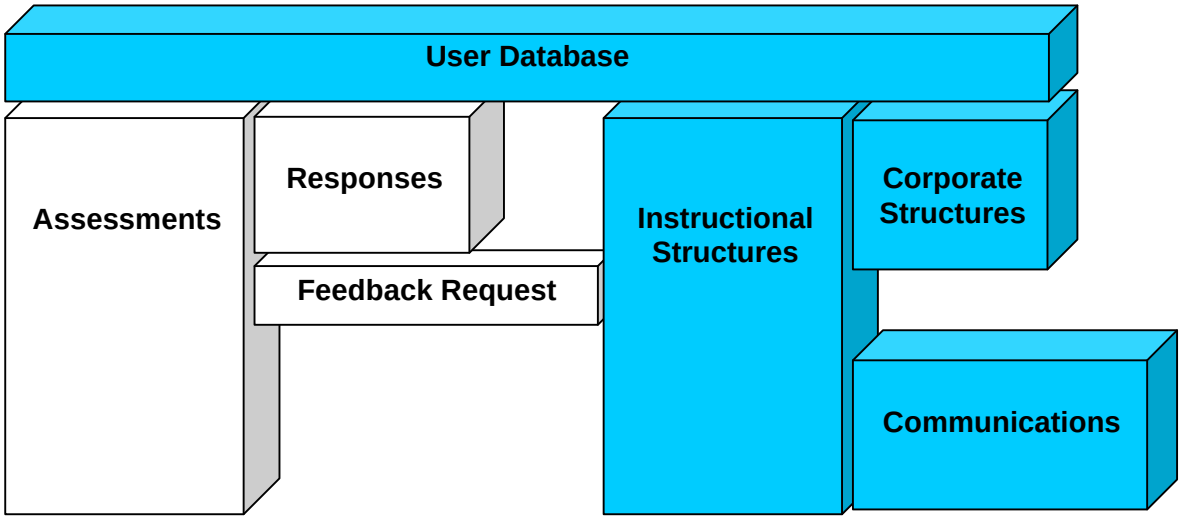


Figure 5-1: System blocks relating to Educational Environments

5.1 Instructional Structures

One of the pitfalls of constructing a data model is creating structures that seem basic and intuitive at the time, but later prove to be inflexible and unable to adequately represent the reality of the human system being modeled. A perfect example of this is the process that I went through in order to model the instructional environments in which assessments are used. On the surface, it seems like a rather simple problem, easily solved by the creation of some sort of a course or subject object. However, my work revealed that creating a single object of type `Course` to try to represent all educational settings is very constricting. For example, if a professor teaches Microeconomics in the Fall of 2003 (Class A), and then again in the Spring of 2004 (Class B), can both of those be said to be modeled as the same course? Not exactly. In reality there is a time variable that creates a difference between the two. This means that there is an element common to Class A and Class B, to which a time variable is added to in order to cause a time-based distinction.

The above example is evidence that to model the structures of content dissemination in a way that parallels the reality of the social systems, a more careful look make be taken to consider what "Instructional Structure" exactly means and what they include. Besides assuring greater model flexibility, taking a measured approach to the system deconstruction can help to avoid semantic confusion caused by the same educational structures terms being used in different places to mean different things.[26]

5.1.1 Defining a Subject

To grasp a better understanding of the principles behind the system, note that the terms "education" or "instruction" imply that there is some sort of content that is being communicated in one way or another. There is an infinite amount of content that exists, so viewing content (or knowledge, facts, understanding, information, etc.) as a whole body is probably not the appropriate place to start. Looking at the social system being modeled, it becomes clear that the structures and nomenclature used in the space (subjects, courses, classes, etc) are ways in which the general mass of content is subdivided into digestible sets of information of various sizes. These sets of information can vary in size from a body of related content to a single fact. For the purposes of our model, the most basic divisions provide the best starting point. Therefore, I codified the concept of "Subject" as follows:

$$\text{Subject} = f(\text{Content})$$

Subject is a function of content, or more specifically, a subject space is defined by boundaries of content, as shown in Figure 5-2.

[26] An interesting example is the term "class" which can be used to talk about:

- An entity offered by an institution on an annual basis (e.g. "this Microeconomics class has been taught by the same professor for 19 years").
- An entity that occurs a single time on a student's transcript (e.g. "I got an 'A' in my microeconomics class").
- An entity that occurs several times a week (e.g. "Wednesday and Friday afternoons I have class, which is Microeconomics.).

It is fairly easy for humans to listen to the above phraseology and determine which entity is being referenced based on context, but (at this time) that is not something that a computer can do, so the data model must make the distinction explicitly.

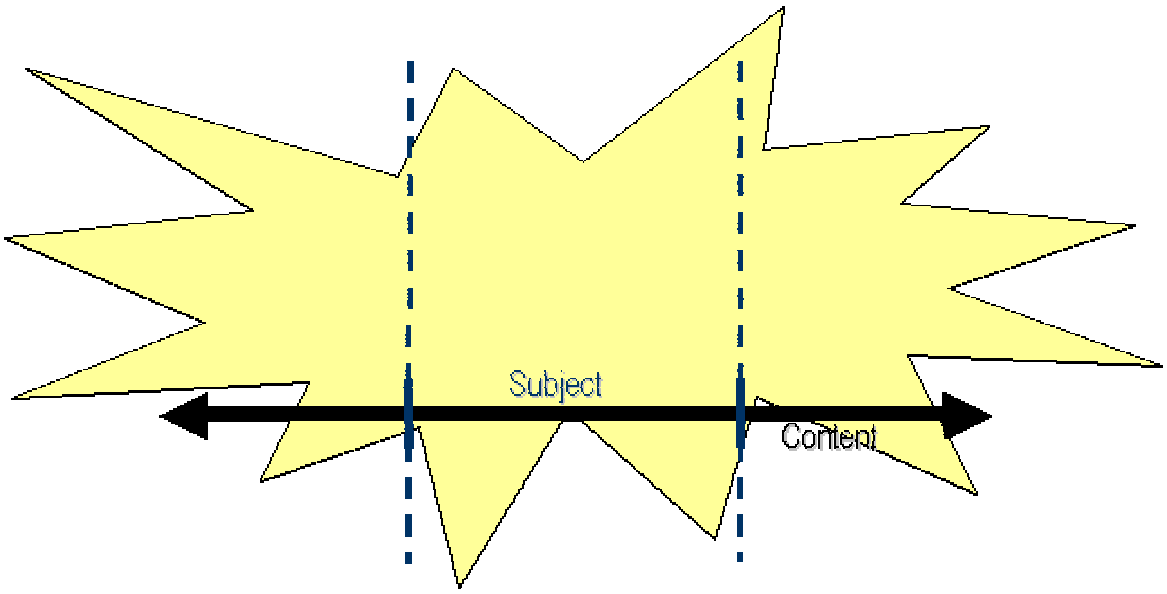


Figure 5-2: Subjects defined on Content Axis

Another way to look at it is to say that content can be visualized as an axis within the global space of all of the instruction which has ever, or will ever, take place. Points of reference along that axis make the classification of instruction much easier. Returning to the previous example, the Subject of Microeconomics is defined by the content that it is considered to contain (or, alternatively, it can be viewed as bounded by content which it does not contain). Therefore, in the previous example, Class A and Class B are both the same Subject (Microeconomics), as they contain the same content.

5.1.2 Defining a Course

However, while Class A and Class B may fall into the same subject, they are still not fully the same. One is given in the Fall of 2003 and the other is given in the Spring of 2004. This highlights a second variable of distinction, that of time, which in this model is used to define the concept of a "Course":

$$\text{Course} = f(\text{Content}, \text{Time})$$

A Course therefore is a function of Content and Time. That is to say, boundaries in time, joined with boundaries on content, define a Course. This concept is visualized in Figure 5-3.

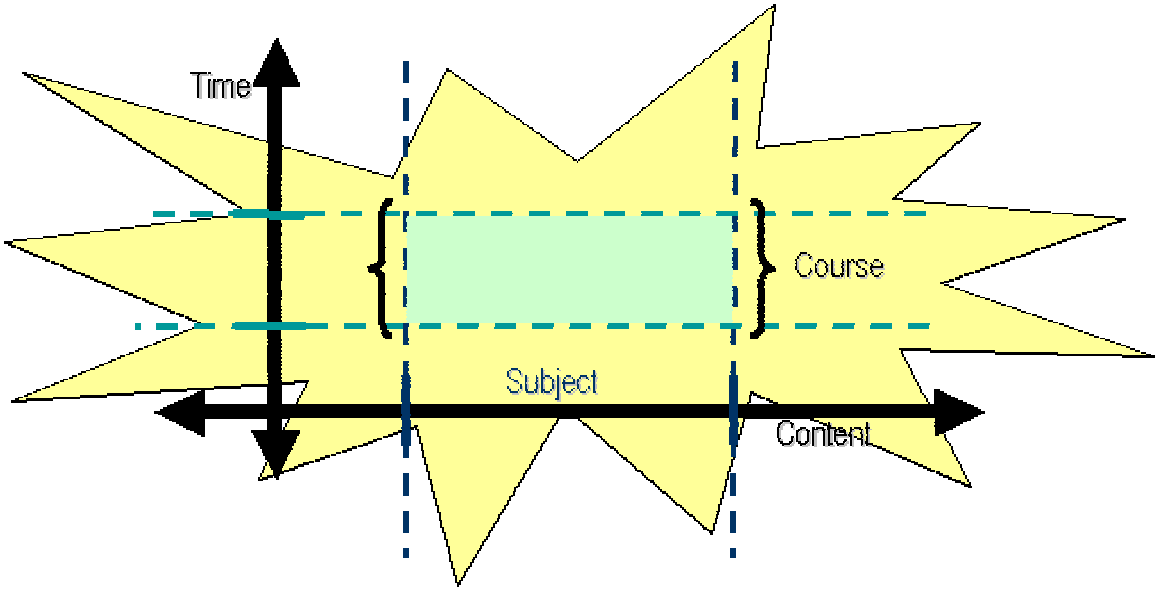


Figure 5-3: Course defined on Time and Content Axes

Applying this concept to the earlier example, Class A and Class B are each the same Subject, but their differentiation in time causes them each to be a different Course.[27] Therefore, by this modeling approach, the intersection of time and content defines a Course.[28]

5.1.3 Defining a Session

There are many applications in which the Course distinction would be of sufficiently small granularity (e.g. applications for tracking student semester grades). However, in order to pursue the system vision of ubiquitous feedback, a finer level of distinction was necessary for this model. If an instructor wished to use a survey every time she lectured, then relating all of those surveys to a single course object would produce an organizational difficulty. Additionally, since some of the use cases depended upon tracking student attendance, it made sense to further divide the Course concept to allow for logical organization of surveys and tracking of student activity on a finer level.

There are two different ways in which a Course might be further subdivided. The first way is based on divisions set along the Content axis. This would create what might be thought of as "topics", or sub portions based on Content. Following the Microeconomics

[27] The decision was made early on to avoid the use of the term "class" within this data model, due to its use to describe many different educational structures in different contexts (see footnote 26 on previous page), coupled with its use within the object-oriented paradigm (a class is a template for an object type). Avoiding the term altogether avoided the confusion around the context of its use.

[28] It is important to note that this is not the only approach for modeling these structures. A third axis marking Location could be superimposed into this system, as well as an axis for Instructor. Every model contains some level of simplifying assumptions (known in signal processing as "quantization error"), and the OnFORME model makes the simplifying assumptions that location and instructors can be adequately modeled as object properties or relationships, as opposed to object types (which would be the equivalent of adding another axis).

analogy, there might be topics like “Intro to MicroEcon,” “Supply and Demand Theory,” “Marginal Cost Applications,” etc.

The other angle on Course subdivision is that of defining smaller blocks along the Time access. This would create what are sometimes called “lectures:” sub portions of a Course based on Time. The Microeconomic example might have lectures like “Monday 3/15,” “Monday 3/22,” etc.

Owing to the fact that not all users will use one or the other of these granularities, and the model needs to be flexible enough to handle both, the design decision was made to create an object which could represent either (or both). This concept of a subdivided Course is defined as a “Session”, where the subdivision could be in Time or Content:

Session = f(Content, Time, [subContent, subTime])

This concept is shown (with both types of divisions represented) in Figure 5-4.

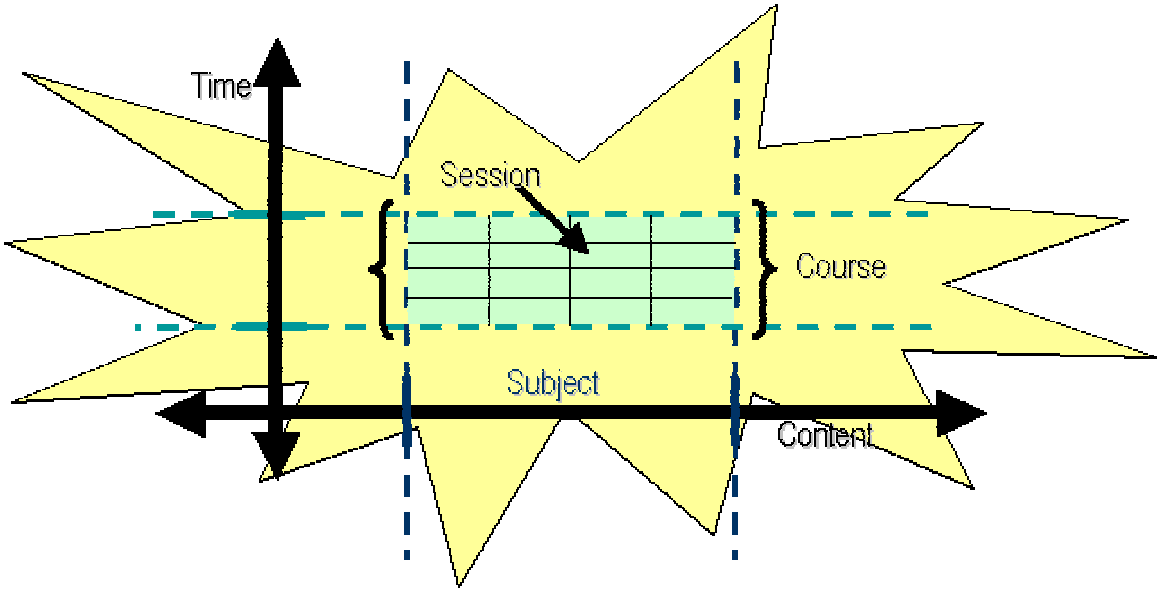


Figure 5-4: Sessions Defined as subsets of Courses

Hence, the concept of “Session” is intended to encompass both the ideas of Topic and Lecture, and provide a way to logically subdivide a Course into whatever sub-units are most appropriate for a given circumstance. Also, note that there are times in which a Course and a Session may be synonymous because a Course consists of only a single Session (e.g. the InterSystems eLearning broadcasts, which are 90 minute one-time events).

5.1.4 Discussion of Object Implementation

Having finalized the preceding classification process, the objects comprising the Instructional Structures block could be implemented. The final class implementation of these concepts is shown in Figure 5-5.

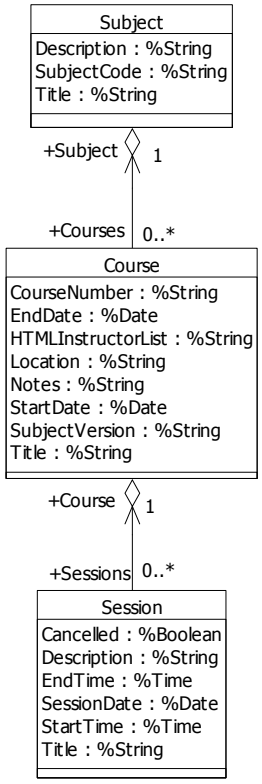


Figure 5-5: Classes contained within the Instructional Structures block

Figure 5-5 is a UML (Unified Modeling Language)[29] diagram that communicates the properties (values) defined within various object classes, and how the classes interact with each other.[30] The diagram reads[31] as follows:

- The `Subject` object has a `Description`, `SubjectCode` and `Title` (all of type `%String`[32]), and it is the parent of `Course` objects, referenced by the `Courses` relationship.

[29] UML is a method of representing object-oriented designs in a graphical way that represents properties, methods, inheritance and relationships. It is one type of data structure visualization.
[30] It is also an option to list the methods (functionality) of object classes in a UML diagram. These have been left out for the purposes of simplifying the diagrams.
[31] This UML diagram translation is included to show the reader how to understand these representations. Further diagrams will not be described in this level of detail.
[32] All class type definitions preceded by a "%" are part of the Caché Object Library.

- The `Course` object is the child of a single `Subject` object (referenced by the `Subject` relationship), and the parent of `Session` objects (referenced by the `Sessions` relationship). Course has 8 different properties: `CourseNumber`, `EndDate`, `HTMLInstructorList`, `Location`, `Notes`, `StartDate`, `SubjectVersion`, and `Title` (all of type `%String` except the two date fields, which are of type `%Date`).
- The `Session` object is the child of a single `Course` object (referenced by the `Course` relationship), and its properties are: `Cancelled` (type `%Boolean`), `Description` (type `%String`), `EndTime` (type `%Time`), `SessionDate` (type `%Date`), `StartTime` (type `%Time`), and `Title` (type `%String`).

The fact that these objects are related through "parent-children" relationships means that the child objects are dependent upon the parent objects.[33] This means that no `Course` can exist without having a parent `Subject`, and no `Session` can exist without a parent `Course`. This structure reinforces the logical reality of the social system, as every course that is taught will be based on a subject, and sessions make no sense outside of the context of a course. This "parent-child" relationship is different from a "one-many" relationship, where the "many" objects can exist independent of the "one" object in the relationship.

This set of three classes provides a simple example of how much easier it is to navigate among data in an object oriented database, as opposed to a relational database. Suppose the application had the record ID for a particular Session (SessionID), and it needed to find the SubjectCode of the Subject to which that Session belongs (since Session is a child of a Course, which is a child of a Subject). Finding that information using SQL (as is used in relational databases) would require code similar to that found in Figure 5-6

```
SELECT Subject.SubjectCode
 FROM Feedback.Session AS Session,
     Feedback.Course AS Course,
     Feedback.Subject AS Subject
 WHERE Session.ID=:SessionID
     AND Session.Course=Course.ID
     AND Course.Subject=Subject.ID
```

Figure 5-6: Finding the SubjectCode for a Session using SQL

This SQL statement is not complicated as far as SQL goes, however it is somewhat convoluted when it comes to representing the intuitive relationships between the various classes in the data model.

[33] In the Caché database, this is translated into child data storage within the parent storage location.

Caché Object-Script offers an alternative approach to accessing data, which is much more intuitive to the native inter-object relationships created in the data model. This approach is called "chaining", and it allows one object to be used as a link to the properties of another. With this approach, once one object of a data model is in memory (or "instantiated"), then any other object within that model can be reached with a single line of code, provided there are relationships at every connection between the first object and the second object. Figure 5-7 shows how a Session object with ID `SessionID` is instantiated (opened) and then used to write the `SubjectCode` of the subject to which it belongs.

```
Set Session = ##class(Feedback.Session).%OpenId(SessionID)
Write Session.Course.Subject.SubjectCode
```

Figure 5-7: Finding the SubjectCode for a Session using Caché Object Script

The above method of accessing data allows Object-Oriented front-end implementation to easily take place and connect with the objects in the back-end database. This allows for end-to-end Object-Oriented development, as opposed to trying to build an OO development methodology on top of a relational data model. The goal of chainable data access was maintained throughout the architecture development process, with the aim of being able to avoid SQL to the greatest extent possible. This speeds up data access, design implementation, and ease of debugging.

Throughout this thesis, there will be many UML diagrams presented to show the implementation of various parts of the object model. These diagrams give a picture of the way in which the objects interact, as well as a listing of the properties in each object class. Many of the properties and relationships are self-explanatory from their names, while some are not. For a complete set of documentation for every object class in the OnFORME application, listing properties, relationships, queries and their use, refer to Appendix C– Software Classes.

5.2 User Database

The second block of the OnFORME system is the User Database block, which contains the classes that model all of the users of the OnFORME system and their interaction with the different parts of the system. The major components of this block are the `Person` class, the `Role` class (and its derived subclasses) and the various many-to-many associative classes used to connect a person's roles with the instructional structure classes. For reference, all of the classes in the User Database are shown in the Figure 5-8.

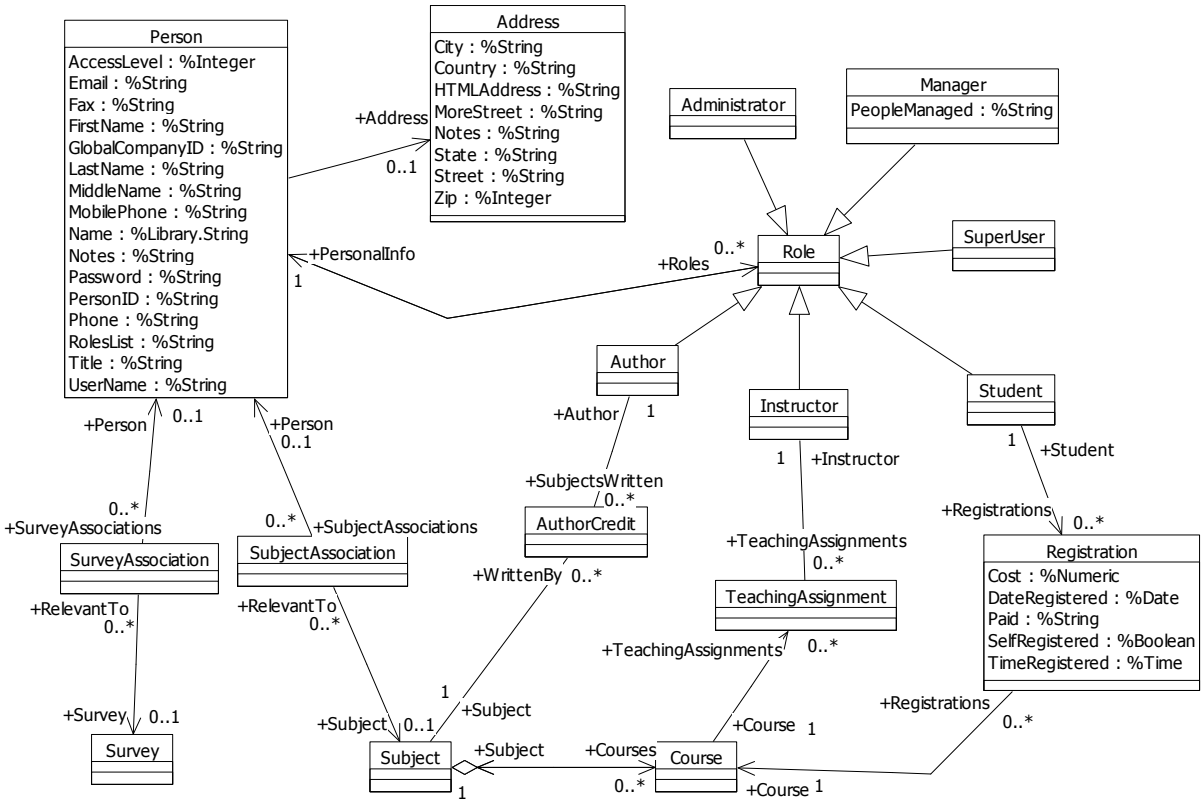


Figure 5-8: Classes contained in the User Database block

5.2.1 The Challenge of Dynamic Polymorphism

In the previous example, modeling the system of instructional structures was somewhat simplified by the fact that the structures being modeled did not change in function after they had been added to the database. For instance, a Session that is added to the database will never need to act like a Subject, because in real life Sessions always remain as Sessions and do not change. Likewise, there was not the challenge of creating objects that must behave as more than one type of item. For example, objects must behave as either a Subject, a Course or a Session, but never as a Subject and a Course, or a Course and a Session. They each retain the single functionality for which they were originally added to the database.

In real life, people in educational systems to not hold to either of these attributes, and therefore modeling system users is considerably more challenging than modeling instructional structures.[34] This is due to the fact that there are a wide variety of different types of people in educational environments. The different people types (or roles) that were included in the OnFORME architecture were:

- `Student` – this is the type of person that will be registered for courses and will respond to surveys

[34] This is a challenge of modeling people in general; it is not unique to the Caché environment.

- `Instructor` – this is the type of person that will be teaching a course and will publish surveys
- `Author` – this is the type of person that creates a Subject and can receive feedback as part of that role[35]
- `Manager` – this is the type of person who is responsible for one or more Instructors and has supervisory privileges over them[36]
- `Administrator` – this is the type of person who manages user rights within the system
- `SuperUser` – this is the type of person who administrates the entire system from a technical perspective

The challenge is that a single person may fit into more than one of these "roles." For example, Person A may be a Student who also serves as a Teacher's Assistant (TA) and therefore needs to be able to function in some regards as an Instructor. If the database contained a Student class and an Instructor class, then it would be possible for an object to contain the properties of both; for example, if a `Student_Instructor` class were defined, it could inherit properties from both the `Student` class and the `Instructor` class. This is called "multiple inheritance," and it is a common Object Oriented (OO) practice. Multiple inheritance allows polymorphism to take place, in which one object can act like multiple other objects. However, with six different user types (and the possibility of adding more in the future), the number of combinations of those types is 63, and a class would have to be created for each combination.[37] In practice, multiple inheritance is not a good approach when there are many combinations that need to be accounted for.

Even if inheritance and polymorphism could be used to account for a person having multiple roles (despite having to create a large number of multiple inheritance classes), the model would still not be able to reflect the reality of people in educational environments. In reality, the roles of users are not static, but they can and often do change. For example, if a person is added to the database as a student, but later becomes a TA, then the `Student` object would have to have the role of `Instructor` added to the original object. With objects representing people in multiple roles, this is not possible without creating a new object, and copying all of the information from the original object.

Implementing user's roles through traditional inheritance and polymorphism is not a practicable option. Therefore, it was necessary to find a way to model the users that allowed their functions and properties to be flexible and dynamic. This would require some sort of "dynamic polymorphism" to properly model the system participants.

---

[35] Feedback to Authors is not implemented in the user interface of OnFORME at this time. The plan is that it be implemented at some point in the future to allow feedback to be sent to instructors about course materials that they created.

[36] The goal of this role is to create a level of authority above instructors, so that feedback that is collected for a number of instructors for performance appraisals, etc, might be visible to a single manager. This functionality is not yet implemented in the user interface of OnFORME.

[37] This is calculated by summing the different combinations of various numbers of roles from 1 to 6, or:
(6nCr1)+ (6nCr1)+…+ (6nCr1) = 6+15+20+15+6+1 = 63.

5.2.2 Initial Failed Attempts

5.2.2.1 "Container" Approach

My first attempt at creating dynamic polymorphism within the model viewed the person as a sort of container, and the roles of that person were blocks that could dynamically be placed inside of, or removed from, that container. The containers were serial classes, which are physically stored inside of the `Person` object.[38] This arrangement is shown in Figure 5-9.

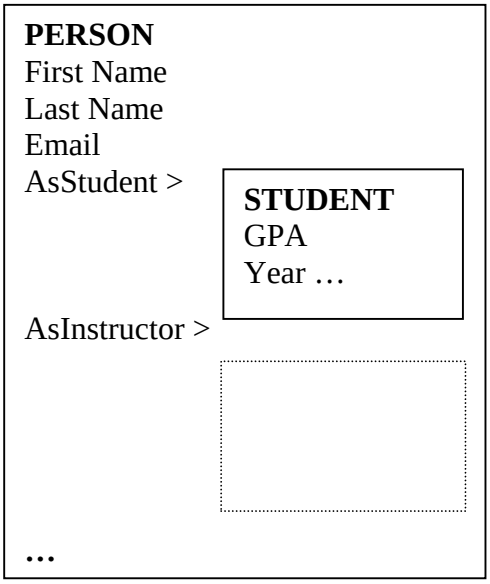


Figure 5-9: "Container" approach to Dynamic Polymorphism

With the container approach, the "AsX" properties would be of serial type X (e.g. "AsStudent" is of serial type `Student`). Therefore, seeing if the AsX properties points to anything could check for the existence of the role. In the example in Figure 5-9, the `Person.AsStudent` property would be a valid pointer (and return a handle), while the `Person.AsInstructor` property would return null (""). Valid pointers indicated that a person object operates in that role, and null pointers mean that they do not.

This implementation contains several issues that make the approach undesirable:

- The different roles that a person might take on are hard-coded into the data structure definition of the Person class. This means that were new roles to be added later, it would require changing the structure of all of the existing person

[38] A serial class is a class without persistent storage. It is for objects that only make sense inside of another object. The serial class is used to group together logical properties, which can then be pulled into another class as a bundle. The `Address` class contained in the User Database block is also a serial class.

objects in the database. This is not difficult to do, but it is a somewhat kludgey[39] approach to incorporating flexibility into a model.

- The Role containers will be pointing to other objects within the database; for instance, a `Student` role would have a `Registration` for a `Course`. Getting from the `Person` to the `Course` would be easy through chaining. However, getting from the `Course` to the `Person` would not be possible without some additional property being created in `Student`, which points back to the `Person` containing that `Student`. This is because the serial class is not a relationship, but a property of `Person`. Relationships have two-way pointers, while properties point in only one direction. Since chaining is a design priority, this is not an option.

The Container approach could not be implemented cleanly, and therefore a new methodology was attempted.

5.2.2.2 "Hook" Approach

To allow for chaining, a new approach was considered using relationships. This approach placed the `Role` objects outside and distinct from the `Person` objects. This approach can be visualized as the `Person` objects having a number of "hooks" (relationships) on which the different roles are placed. Therefore, "AsStudent" would be a relationship with a `Student` object, and the chaining references could go in both directions. This implementation is represented in Figure 5-10.

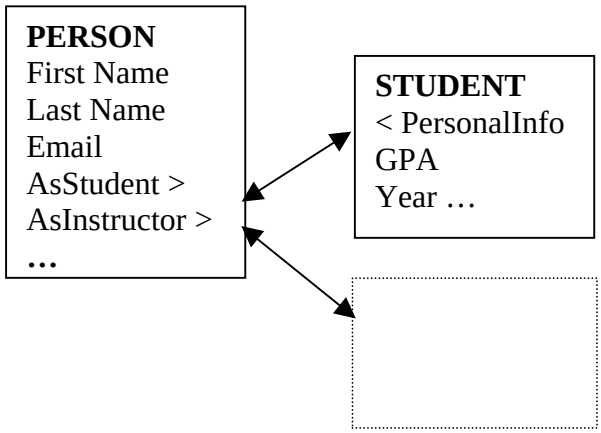


Figure 5-10: "Hook" Approach to Dynamic Polymorphism

The chaining issue has been resolved, because `Person.AsStudent` points to the `Student` object, and `Student.PersonalInfo` points to the `Person` object. This time, there were still three issues that kept this implementation from being as clean as is desirable:

[39] A "Kludge" is a software implementation solution that does not represent clean design and coding techniques.

- As with the "Container" implementation, there is still the issue of the role hooks being hard-coded into the `Person` object. This hard coding does make the `Person` class somewhat kludgey (especially as the list of roles increases and a separate relationship is required for each). This issue causes a messy implementation of the `Person` object, which makes the search for a better solution desirable.
- Using relationships gave rise to an implementation problem specific to Caché. The Caché object layer supports parent-child and one-to-many relationships, but it does not have native support for one-to-one relationships. Were the hooks to be implemented with a one-to-many relationship, the syntax would become unnecessarily complicated. Assuming the `Person` is the "one" and the `Student` role the "many", chaining from Person to Student would be easy and going in reverse would be possible, but somewhat of a kludge, as a relational index is required going from a "many" side to a "one" side. An example of this is shown in Figure 5-11 and Figure 5-12.
- Since a one-to-many implementation would be used, that would also mean that a second student object could accidentally be placed on the hook where one already existed. This would break the model and cause data to become dispersed between parallel objects.[40]

```
Set Property = Person.AsStudent.GPA
```

Figure 5-11: Chaining in from One to Many in "Hook" implementation

```
Set Property = Student.PersonalInfo.GetAt(1).FirstName
```

Figure 5-12: Chaining from Many to One in "Hook" implementation

Since the large number of "hooks" would clutter up the person class and there was not a clean way to implement one-to-one relationships in Caché, a reassessment effort was undertaken to find a better way to implement dynamic polymorphism of user roles.[41] This lead to the final solution: the "Hat Rack" model.

5.2.3 The "Hat Rack" Model

The "Hat Rack" model derives its name from the expression "a person who wears many hats," signifying that they perform many different jobs in different capacities. This model aims to correct for the clutter within the person object caused by a multitude of "hooks" in the previous model, by combining all of those hooks into a single "rack" on

[40] Months after the architecture design was completed, I found out that it would have been possible to enforce only a single role being associated with each "hook", through the use of an index on the `PersonalInfo` property, which is forced to be unique. However, even with this fix, the hook approach is still not clean due to the asymmetry in chaining syntax and the hard coded roles in the `Person` object.

[41] A different implementation would have been desirable even if Caché could support one-to-one relationships, as hard coding the roles is not really "clean" programming.

which the “hats” (or roles), can be placed when they are applied to a person object. The result of this approach is that there is only one `Roles` object associated with each person object, and each discrete role is attached to that object. The analogy is having a single board with hooks on which hats can be hung, as opposed to all of those hooks being scattered around the house.

The implementation of this model uses an Array (called `Roles`), which contains objects of type `Role`. The array is the “hat rack” and each role is a “hat”. The objects placed into `Roles` each inherit from a `Role` prototype class, then have individual properties specific to the role that they are representing. Objects are placed into arrays though the reference of a “key” which points to their placement within the array data structure. Following the hat rack analogy, these keys would be the names of the hats, which are posted above individual hooks (e.g. a “Student” hat, an “Instructor” hat, etc). By checking the hook beneath a label, one can tell whether that person fills that role (in which case a hat, or `Role` object, will be present), or if they do not (no hat). This concept is shown in Figure 5-13.

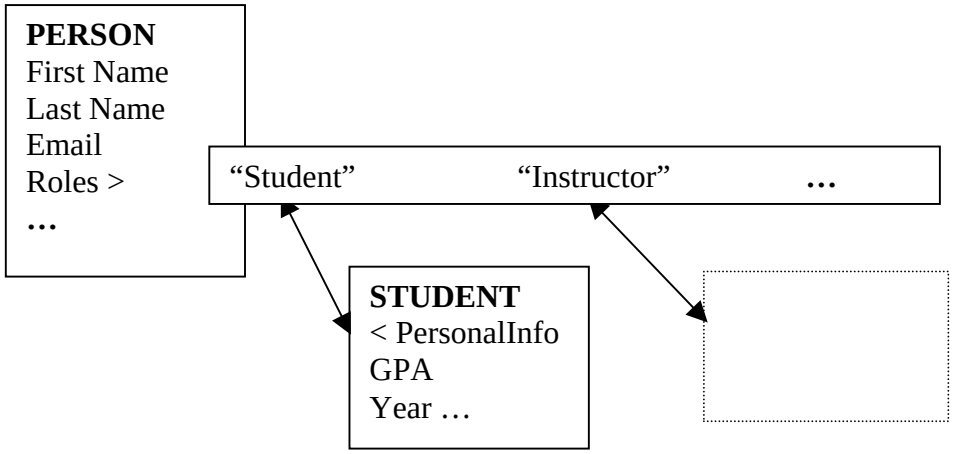


Figure 5-13: "Hat Rack" Implementation of Dynamic Polymorphism

The implementation shown above creates a single place to look for all of the different roles, and it enables new roles to be easily added to the data model in the future as appropriate (e.g. TA, Grader, etc). The code defining the hat rack in the `Person` class is shown in Figure 5-14, which reads as follows: “Create a property called ‘Roles,’ and have that property be an array of type ‘Role.’” This declaration works, as long as every different type of role class (`Student`, etc) inherits from the prototype `Role` class. In this way, every role will act like the `Role` class since they each inherited from it, and therefore they can all be stored in the `Roles` array.

```
Property Roles As Role [ Collection = array ];
```

Figure 5-14: “Hat Rack” declaration in Object Script

The "Hat Rack" model does have one weakness, in that in its basic implementation, there is no way to get from the roles to the person who contains them (as with the "Container" implementation). However, this was worked around by creating a method in the `Person` class that adds the roles to the person object, and can therefore automate the maintenance of the reverse property (`PersonalInfo`). This method is called `AddRole()`, and it received a single parameter of type `%String`, which is the name of the role being added (`Student`, `Instructor`, etc). This method checks to see if a hat exists, then creates a new hat and hangs it on the hook labeled with the name of the role passed to the method. The outcome is code and a logical relationship that is much cleaner than either of the previous two implementations considered, both for chaining and for organization of the `Person` object class. Now, it is merely necessary to check the spot on the hat rack labeled for the role in question, and if it exists, then the properties can be accessed. Examples of chaining in each direction are shown in Figure 5-15 and Figure 5-16.

```
Set Property = Person.Roles.GetAt("Student").GPA
```

Figure 5-15: Chaining from Person to Role in "Hat Rack" Model

```
Set Property = Student.PersonalInfo.FirstName
```

Figure 5-16: Chaining from Role to Person in "Hat Rack" Model

Figure 5-15 logically reads "For the Person object, look at its Roles 'Hat Rack' for the Student hat, and grab its GPA property." This makes sense in terms of both the physical system and the object modeling. Figure 5-16 logically reads "For the Student hat, grab the FirstName of the owner of the 'Hat Rack.'" Again, this is very easy and maps logically to both the actual system and the system representation in the database. Now that the logic and the model have been fully explained, the actual classes shown in Figure 5-17 should make sense.[42]

[42] The Arrow symbols with complete triangles on one end (e.g. from the `Student` class to the `Role` class) signify inheritance.

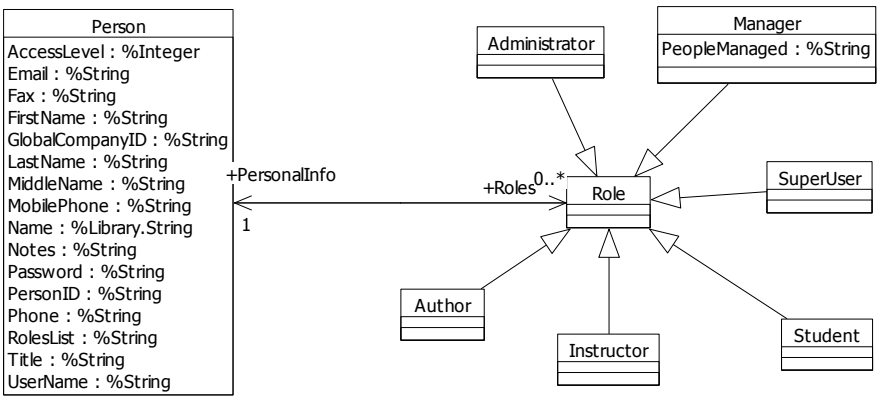


Figure 5-17: Person class and Roles classes

5.2.4 Connecting the User Database with the Instructional Structures

Once the challenge of modeling user roles has been overcome, it was a fairly simple task to connect those user roles with the appropriate parts of the Instructional Structures block. First, it was necessary to recognize that each role would have a many-to-many link with the appropriate part of the content structure. For instance, a `Student` is probably in many courses, and each `Course` has many students. Every intersection between a `Student` and a `Course` needs the ability to store information particular to that student's participation in that particular course (e.g. whether or not they have paid for the corporate use of this system). Therefore, the logical way to structure these connections was through the use of many-to-many associative classes. These classes sit in the middle of a many-to–many relationship, and make a connection between the two outer classes. For example, Figure 5-18 contains a `Registration` class, which sits between the `Student` class and the `Course` class. Each `Registration` points to a single `Student` object and a single `Course` object, thus creating a bridge between them. Therefore, a `Student` has a `Registration` relationship with many objects of type `Registration`, each of which points to a single `Course` object. Likewise, each `Course` has many `Registrations` relationships with many objects of type `Registration`, each of which points to one `Student`. This is the proper way of connecting a many-to-many relationship, and each of these associate classes are shown in Figure 5-18.

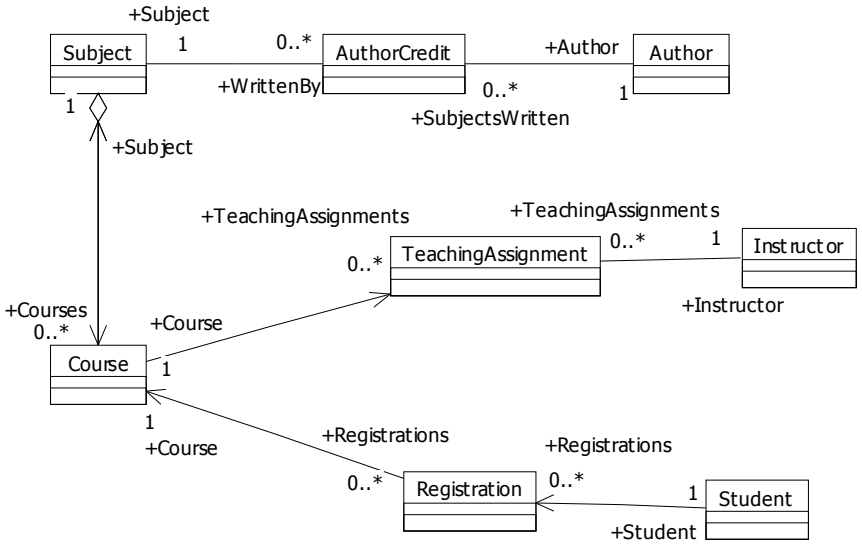


Figure 5-18: Many-to-Many Associative Classes Connecting Users to Structures

The only other class of note in Figure 5-8 is the `SubjectAssociation` class, which is a many-to-many associative class that associates subjects with users. When a user is associated with a `Subject`, then the user can view the subject in the OnFORME interface and interact with it. This provides a way to partition data off from certain user groups, so that their view is localized to those Instructional Structures that concern them.[43]

5.3 Communications

The Communications block is rather simple. Since OnFORME needs to make only mass announcements, there was not a perceived need for a fully-featured communications block. A minimal set of communications features would be sufficient. The architecture was designed so that `Session` objects have a one-to-many relationship with `EmailAnnouncement` objects. The `EmailAnnouncement` objects contain properties that dictate the time they are to be sent, the content of the email, etc. When an `EmailAnnouncement` object is created, it creates a task in the Caché Task Scheduler, which wakes up at the appointed time and date and tells the email to send itself. At that time, the email is sent to the current list of students registered for the parent `Course` of the related `Session`. An `EmailTrigger` class was also created to enable rules-based automated emails (e.g. send an email to all students in a `Course` who have not yet submitted a response to a certain `Survey` object). However, the rules functionality of the `EmailTrigger` class has not yet been implemented. The classes contained in the Communications block are shown in Figure 5-19.

[43] This does not hold true for any people in the system that have only the role of student, because they never see the OnFORME interfaces, merely the surveys which are produced by the system.

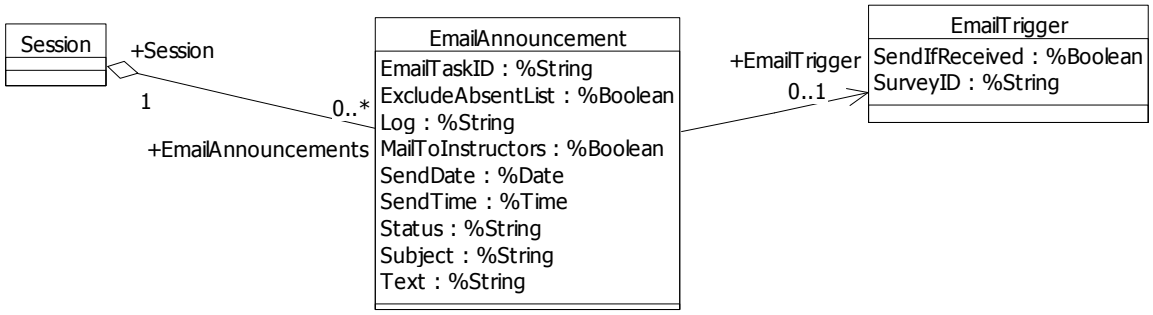


Figure 5-19: Communication Classes

5.4 Corporate Structures

The final block comprising the structural side of the OnFORME architecture is the Corporate Structures block. Most of the classes in this part of the architecture were added much later into the development process. The additions were made to support the business processes in the Learning Services Department that operate in conjunction with feedback collection and course registrations. In the corporate training space, educators also need to track the company to whom their students belong, as well as keep contract information for training performed at customer sites. The `Company` and `OnSiteContract` classes were added to the architecture to allow relevant data to be tracked about InterSystems customers who come in for training. Additionally, an `eLearningSubscription` class has been included for further process modeling efforts.[44] The classes contained in the Corporate Structures block can be viewed in Figure 5-20.

[44] At this point, OnFORME is not supporting the eLearning curriculum of the Learning Services Department. Further work must be done in order to support the business processes currently in place for that program.

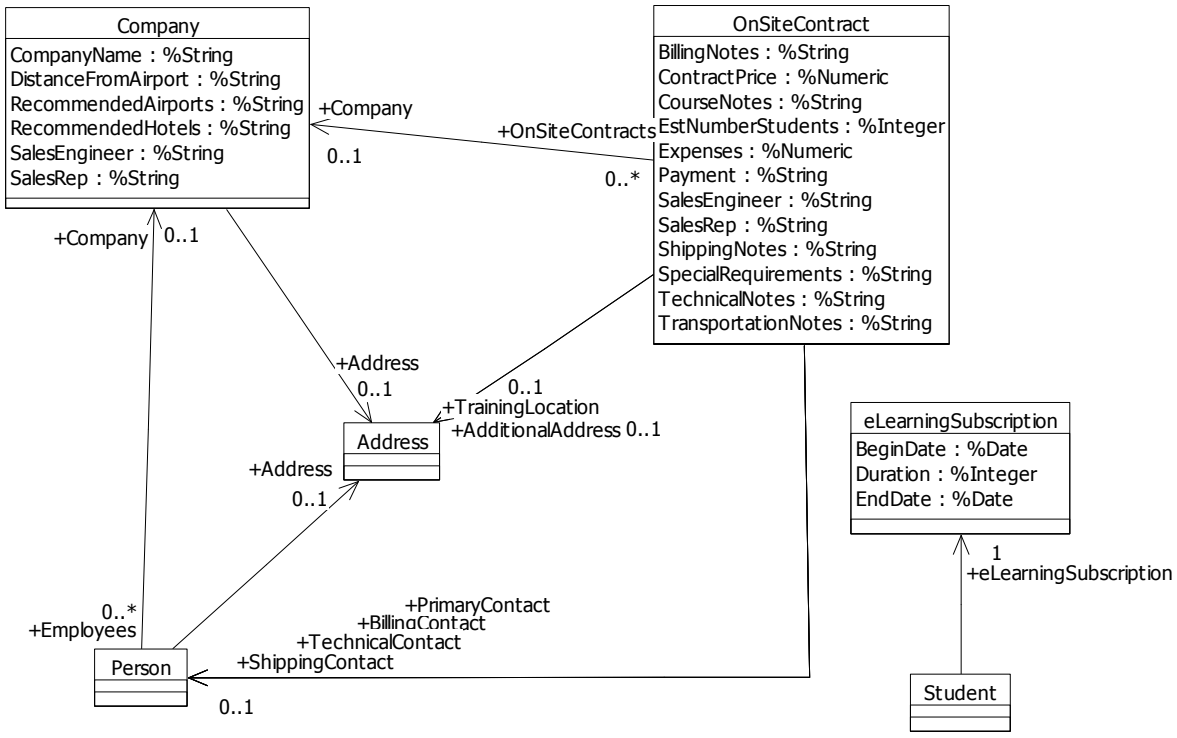


Figure 5-20: Classes contained in the Corporate Structure block

5.5 Summary of Educational Environments

The classes comprising the educational environments portion of the OnFORME architecture are shown in Figure 5-21. These classes create the data structures and functionality that comprise the framework for organizing feedback requests, tracking system users, and communicating to course cohorts. Highlights of this class structure include the decomposition of instructional structures and the dynamic polymorphism employed to properly model the various players in educational settings.

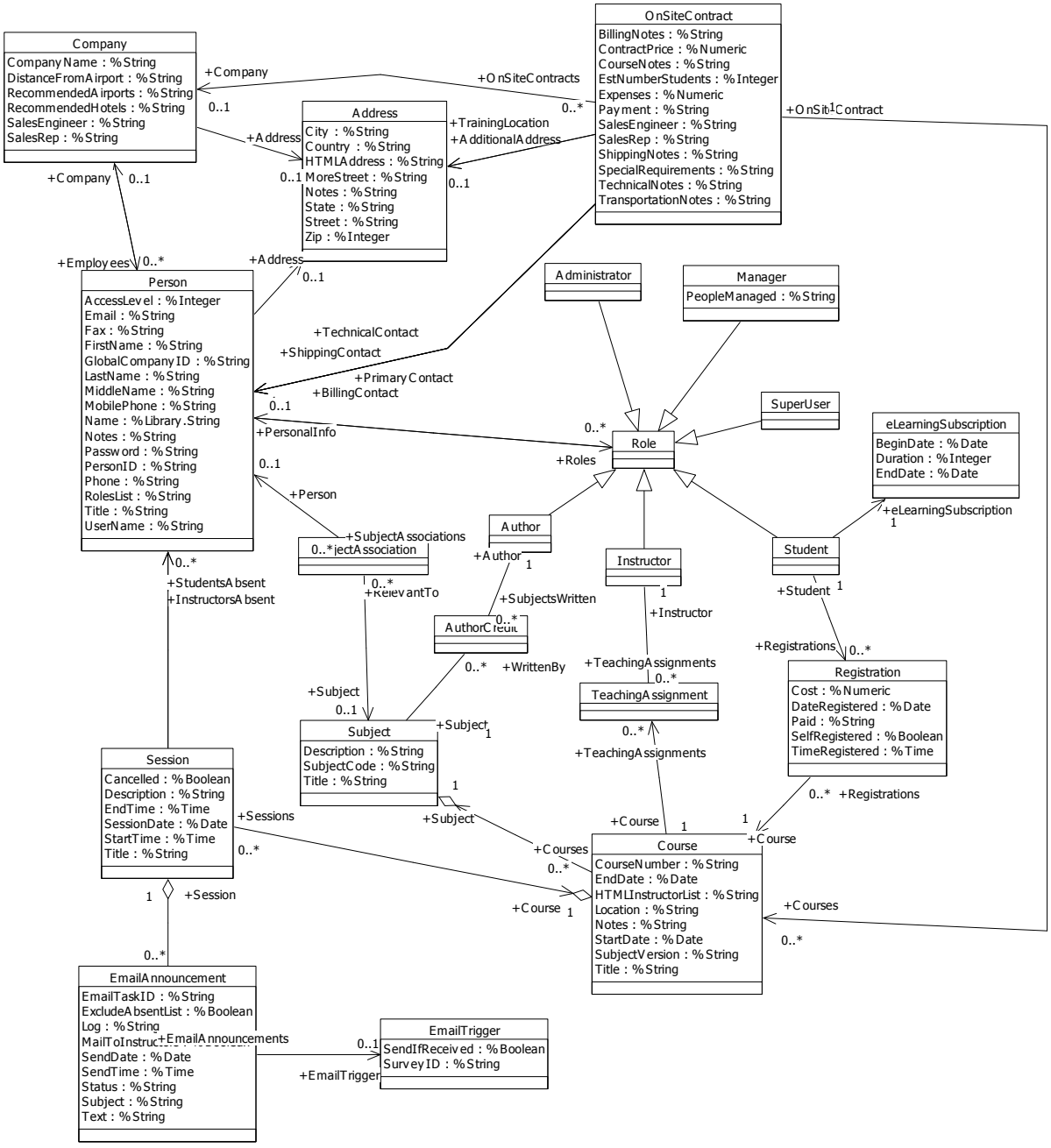


Figure 5-21: Interaction of all Classes Comprising Educational Environments

6 Modeling Feedback Instruments and Their Use

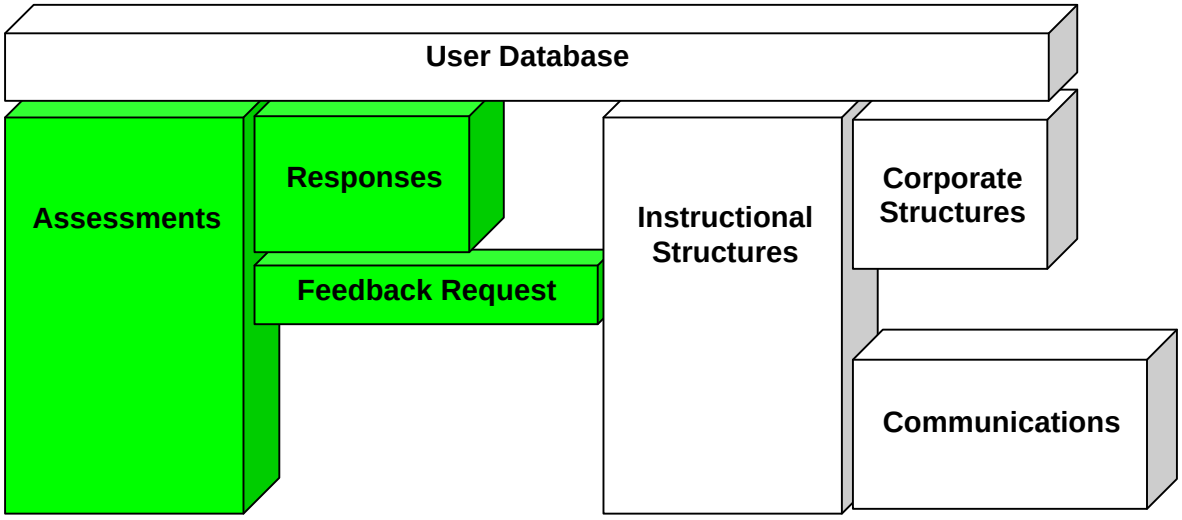


Figure 6-1: System Blocks relating to Feedback Instruments and their Use

This chapter discusses the design decisions made as part of the modeling process of Feedback Instruments and their use. In essence, this part of the architecture is the "feedback engine" which powers the OnFORME system and creates the substantial value to the user. This engine was created around several design goals, which included:

- maximum content reusability
- maximum question type flexibility

The ways in which these goals have been met presents novelty in this application space. Solutions have been implemented in ways rare or unique within feedback applications for educational environments (for more details, see chapter 9 entitled Competitive Analysis).

6.1 Design Goal: Content Reusability

The vision of ubiquitous feedback in educational environments was a driving force behind this project (as well as the pragmatic goals associated with meeting customer needs). In order for that vision to be at all realizable, it was recognized that the barrier of "time investment by instructors" would need to be minimized to the greatest extent possible. Specifically, the time that an instructor spends interacting with OnFORME should be as minimal as possible. This meant that once an instructor created a survey, he or she should never have to re-enter any of that information a second time. Additionally, his colleagues should also not have to enter that same information (provided the original author is willing to share their work). OnFORME is intended to automate the feedback collection process, and a big part of that automation is centered on the easy reuse of all feedback content contained within the system. Most systems provide some level of support for the recycling of surveys (and even questions). However, OnFORME goes a

step further by enabling the reuse of a feedback object itself, rather than simply cloning the object for separate use. This difference will be explained in the following sections.

6.1.1 The Drawbacks of Using Only Feedback Object Cloning

The major of educational feedback systems that exist recognize the need for the recycling of content within a system. The approach used to date by almost all other systems is to allow surveys to be "cloned," so that the content can be recycled for another use.[45] There are also a smaller number of solutions that allow questions to be cloned, adding a finer granularity of content recycling. The OnFORME system included this functionality in its architecture, but it also went a step further, in allowing Surveys and Questions within the system to be "reused" in more than one setting. Here, rather than create a "clone" of the object, a single object can have relationships with more than one "user" of that that object. As an example, questions are used in surveys, and so a "reused" question would have relationships with more than one survey. The details of this are explained in future sections.

The basic content recycling method of "cloning" is certainly feasible from the standpoint of survey authoring, as cloned content works ostensibly the same as "reused" content. If authoring were the only consideration, then it would not make sense to pursue anything further than cloning functionality for feedback objects. However, ubiquitous feedback would require greater considerations beyond simple authoring.

First, the issue arises of maintainability. Assuming that a large number of question and survey objects are created within a system, and that many draw from each others' content (either as a starting point, or as a full content copy), then it is reasonable to assume that errors will be made, circumstances will change, and it will be required to change parts of that content. For example, suppose an instructor wanted to ask a general question of students concerning their overall impression of "Experiment XYZ," which he uses in five different courses each semester. He would create the question once, and then put it in five surveys (one for each course – assuming that the surveys contain questions specific to each course such that the same survey cannot be used for more than one class). Now suppose that after he authors all of the surveys, he realizes that he made a grammatical error in the question referring to "Experiment XYZ," which must be corrected. If the copies of the question in all 5 surveys were "clones" (as would be the case with the other feedback systems on the market), then he would have to edit all five surveys and make the correction. However, if he was using OnFORME and "reused" the question object in the five surveys rather than created clones of it, then he could simply edit the question in one of the surveys. This change would appear in the other four surveys automatically, since each survey is pointing to the same question object. Taken a step further, if the instructor knew that he would teach all five courses for Fall, Spring and Summer term, and wanted to create all of the assessments up front (using each of the five surveys three times), under the cloning paradigm, he would have cloned each survey and made three

[45] Quizlab currently has limited reuse functionality. The Navigo project is the only educational assessment tool that has been identified with a plan towards a model of true reusability. However their full release will not take place until July of 2005. For more details see Chapter 8.2.

distinct copies of it, thus resulting in 15 instances of the cloned question where each would have to be individually corrected. With the OnFORME reuse paradigm, he could reuse the same survey in each offering of the course, and thus correcting the question in a single place would have appropriately corrected all 15 times the question was reused. Thus, reused questions can present considerable time savings over cloned questions when it comes to maintainability.

The second advantage found in the "reuse" structure over the "cloning" structure occurs after the responses have been collected for a survey within a system. If survey content (questions, text, etc) has been authored for single use only, then there is only one way in which the collected data can be reported: giving the responses to the questions for that one use. However, many surveys are not intended for a single use, and in fact they are intended to track progress, which requires use of the same questions and the same survey over a period of time. Under the "cloning" approach to content recycling, a survey would be cloned, and each clone would then receive responses. The results for each instance of the survey would be fairly simple to report by the feedback system, but what about the aggregate results across the various clones of a survey? This is an arduous task, because the results of each of the clones would have to be manually exported and then manually aggregated with the other results from that clone. This is a time intensive process, and it is necessary due to the fact that after a survey's content is cloned, there is no means of correlating the original survey and the other surveys that were created from the original.[46] In the database, each is a separate object that would be reported individually. It is within this area of result set aggregation that the large benefit of object "reuse" occurs. By allowing feedback objects to be reused in more than one place, the original object can be used as a single point of reference when reporting out results. As an example, if the Learning Services Department at InterSystems does an average of two courses per week and they did a customer satisfaction survey at the close of every course, then at the end of the year they would have 104 survey result sets on file. To know their annual customer satisfaction statistics under the cloning paradigm, all 104 result sets would have to be manually exported and aggregated into some common place for analysis. This amounts to nothing less than busy work for some department administrator. However, if, rather than using 104 clones, they reused a single survey 104 times, then all of the results would point to the same survey object, and calculating the annual averages (or averages based on any subset of the aggregate responses for that survey) would be a rather simple task. This same benefit holds true for reusability of individual questions. Drawing from the example in the previous paragraph, with reuse the instructor inquiring about Experiment XYZ would be able to aggregate opinions across all 15 inclusions of the experiment across surveys. However, those results would have to be aggregated and manipulated by hand if question cloning were used.

[46] There is the option of introducing a "cloned from" pointer, however that opens the possibility of a cloned survey changing its content in a way incompatible with that of its prototype (e.g. changing answer options or adding additional questions), which would create synchronization problems between responses that are correlated based on survey. No evidence of "prototype" tracking for clones was found in any of the educational feedback applications reviewed during the course of this project.

6.1.2 Separation of "Data" from "Presentation"

With the paradigm of reuse comes a challenge that does not exist with the cloning approach, and that is the issue of flexibility. When a question is cloned, a copy of all of its attributes is made, and then those attributes can be changed as desired. However, if a question is reused, attributes in the question object will apply to all of the places in which it is used, and therefore changing a property in one use will impact all of the others. This is desirable if correcting a spelling mistake, but not desirable when making a change that is only specific to a single use of the question (e.g. whether or not the question is required). It is therefore necessary to separate out some properties so that they are not reused. These separated properties are made specific to an individual use, while leaving the remaining properties common to all uses in the question object. The challenge is deciding what to separate, and how to separate it.

The type of relational arrangement mandated by object reusability has already answered the "how" challenge. If the goal is to be able to "reuse" feedback objects in more than one place, this means that each feedback object needs to be able to point to many places in which it is used. Additionally, it can be assumed that each place that uses a feedback object (be it a survey or a question) should be able to refer to more than one feedback object. This sets up a many-to-many relationship, which necessitates the use of a many-to-many associative class.[47] This associative class is the ideal place to store properties unique to a single use of a reusable object, as each associative class object represents one "use," or connection, between each of the objects with which it connects. Therefore, properties specific to a use can be stored in the associative class, while properties common to all uses can be stored within the reusable object.

This leads to the question of "what" should be separated out of the reused object. This is a question to which there are many possible answers. The OnFORME architecture uses a dividing line between "data" and "presentation" to determine what goes where. Simply stated, "data" is the content of a question entered by a professor. For a Question object, this would include the question prompt, the number of answer options, the answer option text, whether or not to include a free form answer, etc. For a `Survey` object, this would include the `Question` objects that it contains. Alternatively, the "presentation" properties dictate the way in which an object is viewed in a client browser, and the rules governing its proper use. For a Question object, this might include whether or not it's required, the font size, how to display the answer options, it's position on the page, etc. For a Survey object, this might include the access rights, the dates the survey is available, the behavior after a survey has been completed, etc.

This distinction between "data" and "presentation" is an important one if the content that is manually entered by authors of feedback objects is to be leveraged to its greatest potential. On a higher level, when someone asks someone else a question, it is the substance (or "data") within that question that warrants a response. The respondents should not respond to the way in which it is asked, or the structure of the asking ("presentation"), but rather they respond to the substance of the question and the

[47] For a description of the many-to-many associative class, refer back to section 5.2.4.

substance of the options from which they may choose an answer. Therefore, this data part of the question represents the true value of the question and the central idea around which a body of responses can be built. By stripping it of its presentation accoutrements, it is possible to enable the unique substance of that question to be used in more than one place. This is accomplished while allowing the way in which it is used (or presented) to be chosen and specified on a case-by-case basis. Reusability through data/presentation separation is one of the most powerful concepts employed in the OnFORME architecture, due to how it can be leveraged in the reporting and analyzing of response aggregation. With these principles clarified, this thesis will now proceed to explain how these concepts were realized within the design.

6.2 Assessments

The classes comprising the Assessment block are shown in Figure 6-2. The four major classes (and their child classes) discussed in detail are:

- `Survey`
- `Slot`
- `Entry`
- `AnswerOption`

The other class of minor interest in this block of OnFORME is the `SurveyAssociation` class. This class is similar to the `SubjectAssociation` class discussed in the last chapter, in that it is a many-to-many associative class used to give survey access rights to various users of the OnFORME system.

The following sections discuss the classes listed above, their subclasses and how they fit together to meet the design goals of the system.

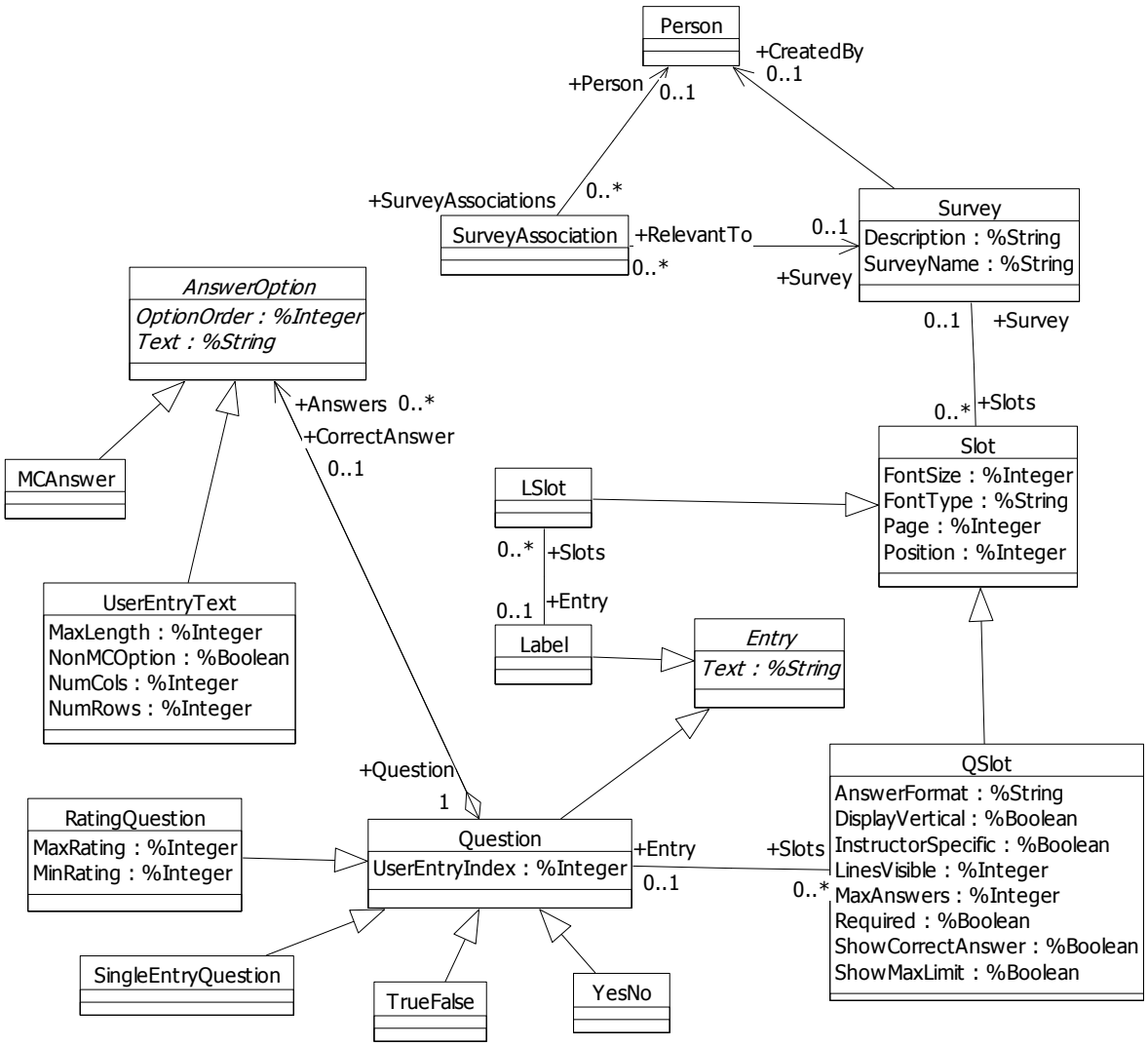


Figure 6-2: Classes Comprising Assessments

6.2.1 Survey

The `Survey` class is the top-level object that ties together the different parts of an assessment. Its function is that of a collector - it points to the different objects included in the assessment. Specifically, a `Survey` has a one-to-many relationship with multiple `Slot` objects. The `Survey` object also contains a property denoting its name and description – both of which are for record keeping only (they are not shown to those taking the survey). Additionally, each survey has a `CreatedBy` property which points to the `Person` object that was the original author of that `Survey` object. It is important to note that `Survey` objects cannot be responded to until they are related to a `Session` object (details of this will be explained later). With this implementation, `Survey` objects require a context in order to be used.

6.2.2 Slots

The `Slot` class is a prototype class for the two different kinds of slots that actually exist within a survey. There are slots for `Question` objects and for `Label` objects.[48] The concept behind the `Slot` class is that each survey is comprised of a number of different objects, each of which is contained in a slot that has a certain display order (controlling the order of the objects displayed in a survey). This is the first example of the "reuse" principle in implementation. If a `Question` object is to be used in more than one `Survey`, it makes sense that it will probably appear in a different place in each survey. Therefore those two different pieces of `Position` data need to be stored in the many-to-many associative class. In this case, that many-to-many class inherits from the `Slot` class.

There are two different types of slots that are contained in a `Survey` object. The first is for `Label` objects, which represent entries that are informative blocks of text within a survey (e.g. section headers, instructions, etc). The second is `Question` objects, which are interactive entries in the survey, intended to collect data from the survey respondent. The `Slot` class contains properties that hold the presentation data relating to:

- the `Position` of this slot in the survey
- the `Page` on which this slot appears in the survey [49]
- the `FontSize` in which the text of this slot should be rendered in the browser
- the `FontType` that should be used.[50]

Since the `Slot` class is a prototype of the two classes that are used in practice, those two classes each contain these four properties by default.

Apart from the properties of these classes (described below), there is another feature of note that is central to the concept of slots within a survey. Since the presentation layer attributes belong to the slot classes, the functionality for rendering that presentation in the browser also is found in the slot classes. Both of the classes that inherit from `Slot` contain a `RenderHTML()` method, which outputs the contents of the object contained in the slot as formatted HTML and JavaScript. This formatting includes the appropriate presentation attributes, as well as the code necessary to enforce behavioral requirements (e.g. required questions, max number of answers, etc). Putting the rendering method within the `Slot` class (as opposed to within `Question` or `Label` classes) was the logical way of implementing reusable survey entries, as the `Slot` object contains the use specific parameters.

---

[48] A "prototype" class is not actually instantiated in the database, rather it is used to define a class structure which is inherited from by subclasses that create objects in the database.

[49] Multi-page surveys have not yet been implemented in OnFORME, but this property has been included in the architecture in anticipation of a future enhancement enabling multi-page surveys.

[50] FontType implementation was not a high priority of the initial customers so it is currently not pushed through to the user interface. However, adding it to the system would be a trivial task.

6.2.2.1 LSlot

The `LSlot` class is a many-to-many associative class used to relate `Label` objects to `Survey` objects. Labels are non-interactive blocks of text used for explanation and description within a survey. By having the `LSlot` class act as an associative class, `Label` objects can be reused among different surveys, where each survey uses the exact same `Label` object in the database, rather than clones. All of the `LSlot` properties are inherited from its `Slot` superclass, and all of these properties represent the presentation information relevant to that particular `Label` object.

6.2.2.2 QSlot

The `QSlot` class is a many-to-many associative class used to relate `Question` objects to `Survey` objects. Questions are interactive entries in the survey that are used to solicit information from the survey respondent. By having the `QSlot` class act as an associative class, `Question` objects can be reused among different surveys, where each survey uses the exact same `Question` object in the database, rather than clones. The `QSlot` class inherits four of its properties from its `Slot` superclass, and then proceeds to define several others that relate to `Question` objects, but not `Label` objects. These properties are:

- `AnswerFormat` – This property determines the presentation of the answer options for a question. This property can only take on one of four values, and the option selected dictates the way that the question is rendered in the browser.
    - `Radio` – the answer options have radio buttons, and only one can be selected at a time
    - `Option` – the answer options have check boxes, and more than one can be selected at a time
    - `Dropdown` – the answer options are displayed in a dropdown box, and one or more can be selected at a time
    - `UserEntry` – there is a single text box in which the respondent can type an answer
- `DisplayVertical` – This is a Boolean[51] property which, when set to true, causes the answer options associated with a question each be on a separate line. Otherwise, all answer options are displayed without intermediate line breaks.
- `InstructorSpecific` – This Boolean property allows the user to leverage the strengths of question reusability. When this property is set to true, the question becomes context aware and self-adapts to the setting in which it is used. When it is rendered in a web browser as part of a survey, it retrieves the list of instructors registered to teach in the course where the survey is used, and repeats the question with specific reference to every registered instructor. Then the results are tagged as instructor specific, which means only the instructor for whom a response is given can view the response in the OnFORME interface. For example, if a question states “Please rate the

[51] A Boolean property is one that can only take on the value of “true” or “false”.

clarity of the instructor's lectures" and is the `QSlot` of the `Question` object has the `InstructorSpecific` flag set to true, then if the course in which the survey is used has two instructors registered (suppose a professor and a TA), then that single question will be rendered twice. The responses to that question will be marked so that they can be viewed only by the instructor designated in that particular rendering of the question (and system administrators). This concept of dynamic questions which adapt to the courses in which they are used has not been seen anywhere else in this application market.

- `LinesVisible` – This property determines the presentation of the answer options when the presentation is selected to be of type "Dropdown". This determines the number of options to make visible within the dropdown box.
- `MaxAnswers` – This property contains the number of answer options that can be chosen for questions of type "Option" or "Dropdown". This upper limit is enforced through JavaScript when the question is rendered in the respondent's browser.
- `Required` – This is a Boolean property which signifies whether or not a question is required. When it is required, the survey cannot be submitted without an answer to that particular question.
- `ShowCorrectAnswer` – This property is not yet implemented, but it will be used in the future to signify whether or not to show the respondent the correct answer to this question following a submission.
- `ShowMaxLimit` – This is a Boolean property that displays a prompt telling the respondent how many answer options can be chosen for that particular question.

All of the properties stored within a `QSlot` object are rendered in the respondent's browser when the `RenderHTML()` method is called that a particular `QSlot` object.

6.2.3 Entry

Just as the `Slot` super-class is the representation of the Presentation layer, the `Entry` super-class contains the Data layer. The `Entry` super-class therefore the actual content of the items within a survey. The `Entry` class is an abstract[52] super-class for the `Label` and `Question` classes, each of which have a one-to-many relationship with their respective `Slot` subclass. The class structure enabling many-to-many relationships between surveys and the items they contain is shown in Figure 6-3.

[52] An abstract class does not have storage in the database, and can only be used for modeling a class structure for classes that inherit from it.

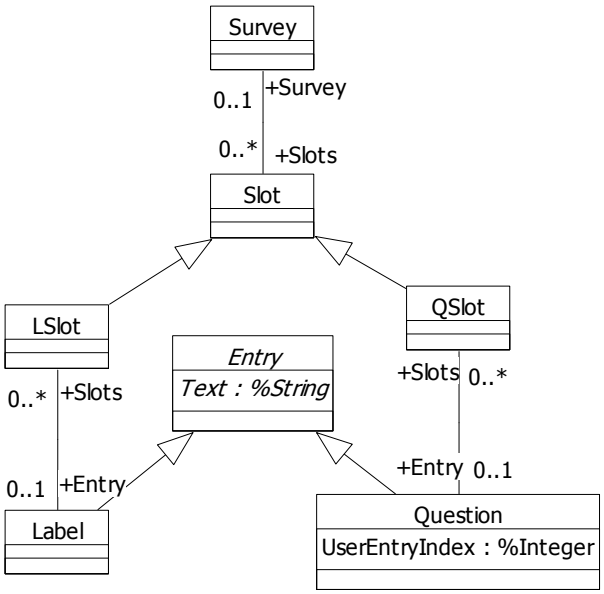


Figure 6-3: Inheritance Structure for Slots and Entries

By creating a common super-class for labels and questions, `Entry` objects can be handled with common code in the parts of the application that do not care what kind of entry object is being manipulated. This allows for code reusability that would not be possible if each type of survey entry were distinct in their super-class definition.

The `Entry` class contains a `Text` property that contains the actual "data" which is reused in all `Entry` objects.

6.2.3.1 Label

`Label` objects are simply text entries within the survey. They have no properties beyond the `Text` property they inherit from the `Entry` class. When it is used in a survey, it is related to an `LSlot` object, which holds the presentation data applicable to how it should be displayed (font size, position within survey, etc).

6.2.3.2 Question

`Question` objects are the interactive entries within the survey. They are composite objects, in that they contain an instance of the `Question` class, as well as `AnswerOption` objects that relate to the `Question` object (described in the next section). The `Question` class inherits the `Text` property from the `Entry` class, and it also has an optional field which points to the `UserEntryText` object that it contains (see following section). The `Text` property contains the question prompt, i.e. the text used to tell the respondent what is expected for that question (e.g. "please choose one of the following", or "please explain"). When it is used in a survey, it is related to a `QSlot` object, which holds the presentation data applicable to how it should be displayed (font size, position within survey, answer options display settings, etc).

6.2.4 AnswerOption

Each `Question` object points to one or more `AnswerOption` objects. These are discrete options between which the user can choose when responding to a particular question. The `AnswerOption` class is an abstract super-class for the two types of answer options that are actually stored in the database – multiple choice options and user entry textboxes. Every `AnswerOption` object has the following two properties:

- `OptionOrder` – This property contains the order in which this answer option should be rendered in the browser. For textbox answer options, this value is set to 0, and it is rendered after each of the other answer options.
- `Text` – This property contains the text displayed for this answer option.

The two types of answer options (discussed below) inherit these properties from the `AnswerOption` class.

6.2.4.1 MCAnswer

The `MCAnswer` class is used for multiple-choice answer options. The respondent can select answer options when they are rendered in a browser. The response collected for a question comprised of only `MCAnswer` objects would be a list of the `MCAnswer` objects selected.

6.2.4.2 UserEntryText

The `UserEntryText` class is used to collect free-text data from survey respondents. These answer options can collect and return entered text when rendered in a browser (e.g. a "please explain" textbox). When text is entered into a `UserEntryText` entry in a survey, the response returns the `UserEntryText` object as selected, as well as the text entered for that option. In addition to those properties inherited from the `Entry` class, the `UserEntryText` class has the following properties:

- `MaxLength` – This property determines the maximum length of the text that can be entered in is textbox field.
- `NonMCOption` – `UserEntryText` objects default to being displayed in the same way as the `MCAnswer` objects, as dictated by the `AnswerFormat` field in the `QSlot` object. Setting this Boolean field to true causes the text option to be rendered separate from the display settings chosen for the other answer options (see example below).
- `NumCols` – This property holds the width of the rendered textbox.
- `NumRows` – This property holds the height of the rendered textbox.[53]

[53] Post-design analysis revealed that having the `UserEntryText` class hold the `NumCols` and `NumRows` properties was a slight departure from the separation of data from presentation (as `UserEntryText` objects are part of the data side of the architecture). However, due to the lateness of this discovery, and the higher priority of implementing other changes, the original architecture was not changed to move these fields to the presentation side of reusable objects.

6.2.5 Question Examples

At this point, a couple of examples would help to put all of the pieces together.

6.2.5.1 Example 1 – Single Selection Question

Suppose a question asked "What color is the sky?" and the survey author wanted to have the options of "blue", "orange", and "other", where "other" could collect text, and only one option could be selected. The desired question format is shown in Figure 6-4.

Figure 6-4: Single Selection Question Example

This question would have 2 `MCAnswer` objects ("blue" and "orange"), and 1 `UserEntryText` object (whose `Text` property equals "other"). The `AnswerFormat` field would be "Radio", and the `UserEntryText` object's `NonMCOption` property would be set to false (which makes the textbox include a radio button). Since the question is being rendered with radio buttons in the browser, only one answer option can be chosen at a time, and so the `MaxAnswers` property does not need to be set in the `QSlot` object. The object representation is shown in Figure 6-5.

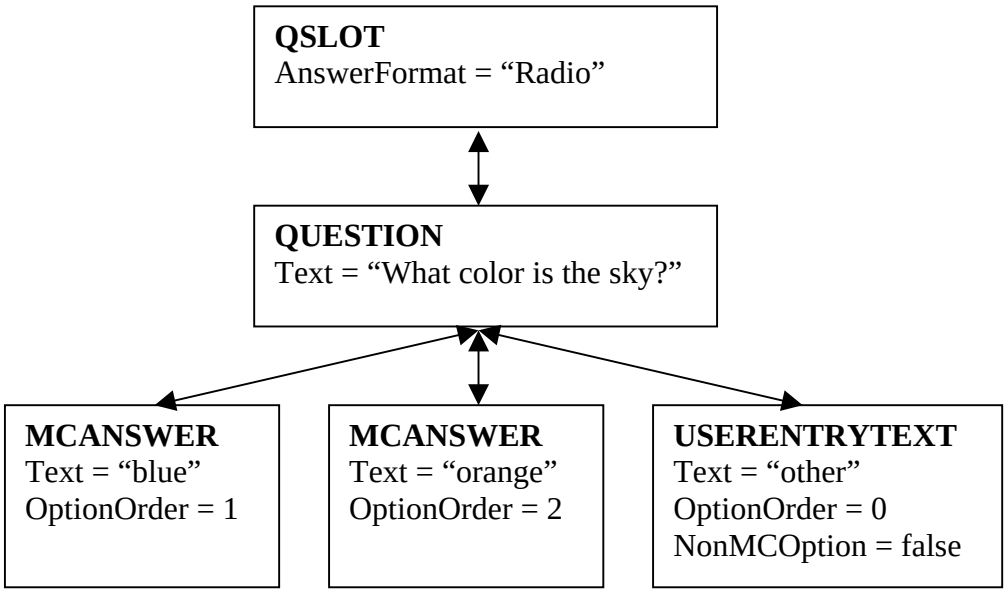


Figure 6-5: Objects contained in Example 1

This example question is a composite of 4 different objects holding the question data, and 1 object containing the usage presentation information.

6.2.5.2 Example 2 – Multiple Selection Question

As a second example, suppose a question asked "Where have you seen our ads? (check all that apply)" and the survey author wanted to have the options of "tv" and "radio", as well as a textbox that prompts "comments on the ads:". The desired question format is shown in Figure 6-6.

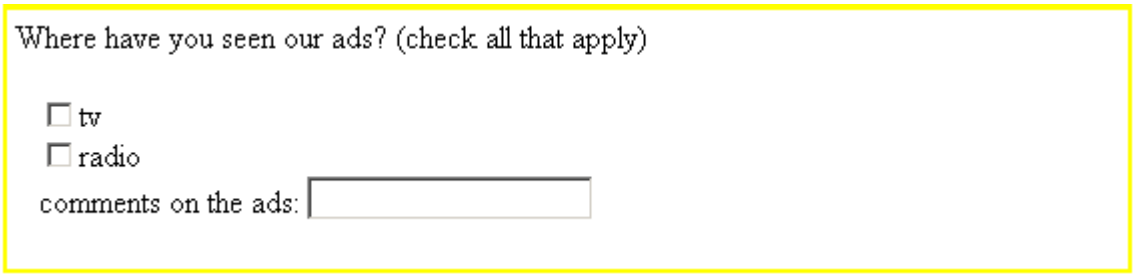


Figure 6-6: Multi-Selection Question Example

This question would have 2 `MCAnswer` objects ("tv" and "radio"), and 1 `UserEntryText` object (whose `Text` property equals "comments on our ads:"). The `AnswerFormat` field would be "Option", and the `UserEntryText` object's `NonMCOption` property would be set to true (which makes the textbox not include a checkbox). The `MaxAnswers` property would be set to '2', thus allowing a respondent to select both check boxes. The object representation is shown in Figure 6-7.

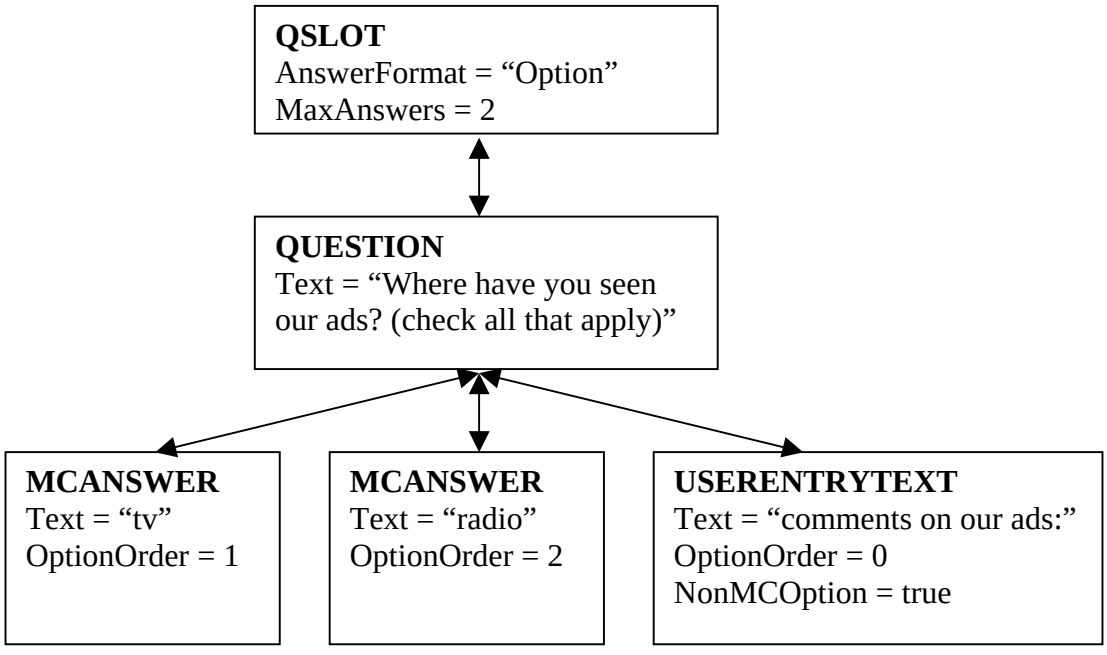


Figure 6-7: Objects contained in Example 2

This example question is also a composite of 4 different objects holding the question data, and 1 object containing the usage presentation information.

6.2.5.3 Example 3 – Text Box Question

As a last example, suppose a question asked "Please tell us what you would like to see improved?" where the respondent was greeted with a single text box. This question format is shown in Figure 6-8.

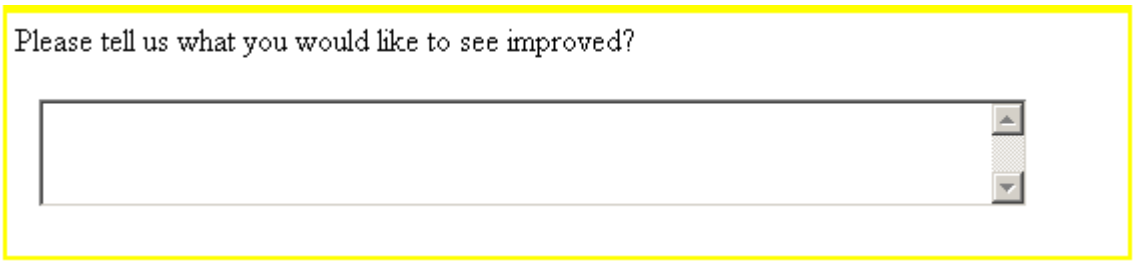


Figure 6-8: Text Box Question Example

This question would have a single `UserEntryText` object (whose `Text` property equals "Please tell us what you would like to see improved?"). The `AnswerFormat` field would be "UserEntry", and the `UserEntryText` object properties are set to create a larger textbox. The object representation is shown in Figure 6-9.

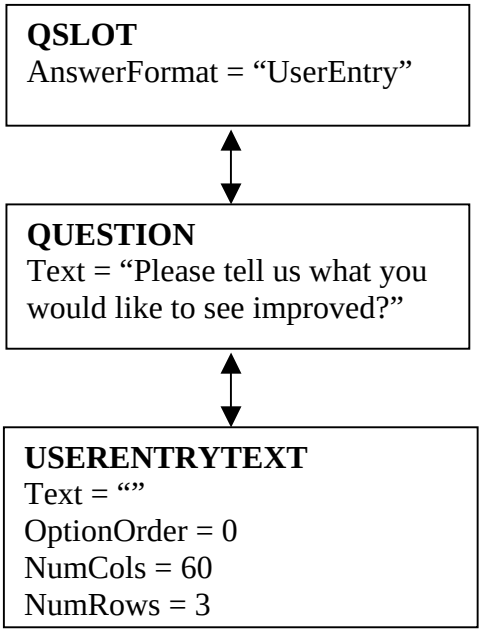


Figure 6-9: Objects contained in Example 3

This example question is a composite of two different objects holding the question data and one object containing the usage presentation information.

The above examples show simple ways in which `AnswerOption` and `Question` objects can be combined to create questions in a survey.

6.3 Responses

Aside from the survey objects, the next most important parts of the feedback collection process are the classes that comprise the responses to surveys. There are two different classes that make up responses; these are the `Response` and `Answer` classes, which are shown in Figure 6-10.

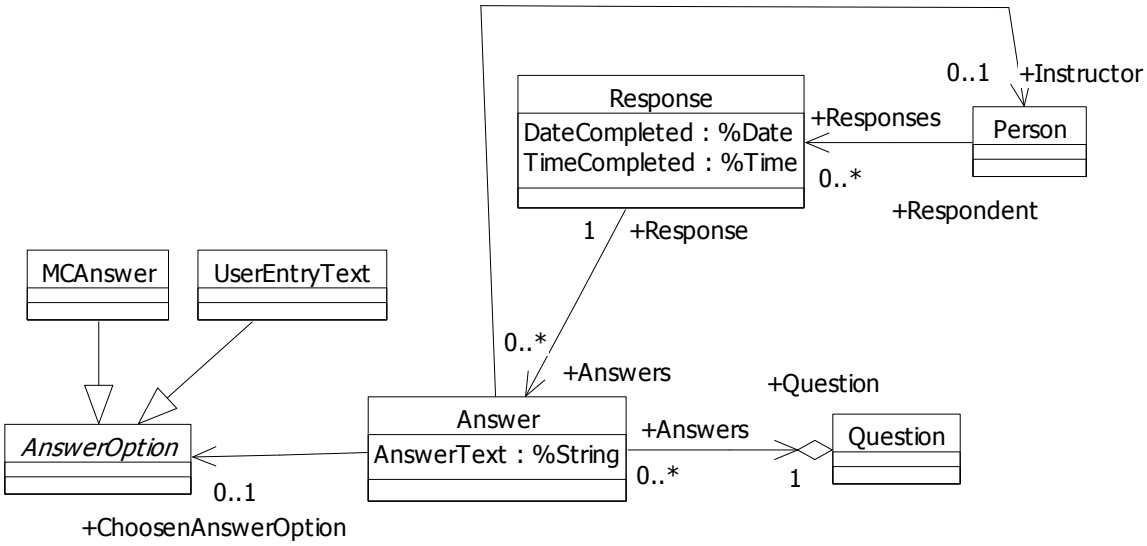


Figure 6-10: Classes comprising Survey Responses

6.3.1 Response

The `Response` class is the centerpiece of the data collected from respondents. This class automatically logs the date and time that the response was submitted (which translates into the date and time that the `Response` object was created). It also has a many-to-one relationship (`Respondent`) with the `Person` class, pointing to the user who submitted that particular response. Additionally, it points to a collection of `Answer` objects, which contain the actual data submitted by the respondent.

6.3.2 Answer

Each `Answer` object stores up to three pieces of information:

- `Question` – This is a many-to-one relationship with the `Question` class, which points to the `Question` object for which this `Answer` object was collected.
- `ChoosenAnswerOption` – This is a property that points to an `AnswerOption` object, which can be either a `MCAnswer` object or a `UserEntryText` object. The object referenced is the one that was selected for the `Question` object referenced above.
- `AnswerText` – If `ChoosenAnswerOption` points to a `UserEntryText` object, then the respondent had the option of entering text as part of the answer. The `AnswerText` field is a property contained in the `Answer` class, which holds the text inputted by the respondent.

In the database, a user's response is the aggregate of a single `Response` object, and 0 to N `Answer` objects. The decision was made to store the responses as objects pointing to answer options, as opposed to copying all of the survey data into an object and marking selected responses, because the former is a much more efficient way of storing the necessary information without redundancy.

6.4 Bringing it all together: the FeedbackRequest

This section discusses the `FeedbackRequest` class, why it is needed, and what is enabled by its implementation.

6.4.1 FeedbackRequest Rationale and Implementation

As stated earlier in this thesis, one of the major goals of the OnFORME architecture is maximum reusability of the content contained within the system. The previous sections discussed the ways in which `Question` and `Label` objects can be reused through the use of the `Slot` objects. The reuse principle equally applies to the reusability of surveys within the OnFORME system. As before, reuse requires a many-to-many associative class that serves as a connector between a survey and the place where it is used.

The connection between the educational structures and the assessments occurs with the `Session` class. The logic is that since the sessions are the objects of finest granularity within an educational environment, this is the best point to which surveys should be linked.

The many-to-many associative class that connects a `Survey` object to a `Session` object is the `FeedbackRequest` class. This represents a discrete "use" of a `Survey` object within the context of a `Session`. In addition to making the connection between a `Survey` and a `Session`, the `FeedbackRequest` class is also the point of reference to all `Response` objects collected for this particular survey "use". In so doing, the `FeedbackRequest` class ties together three of the major sections of the architecture. The relationship between these classes is shown in Figure 6-11.

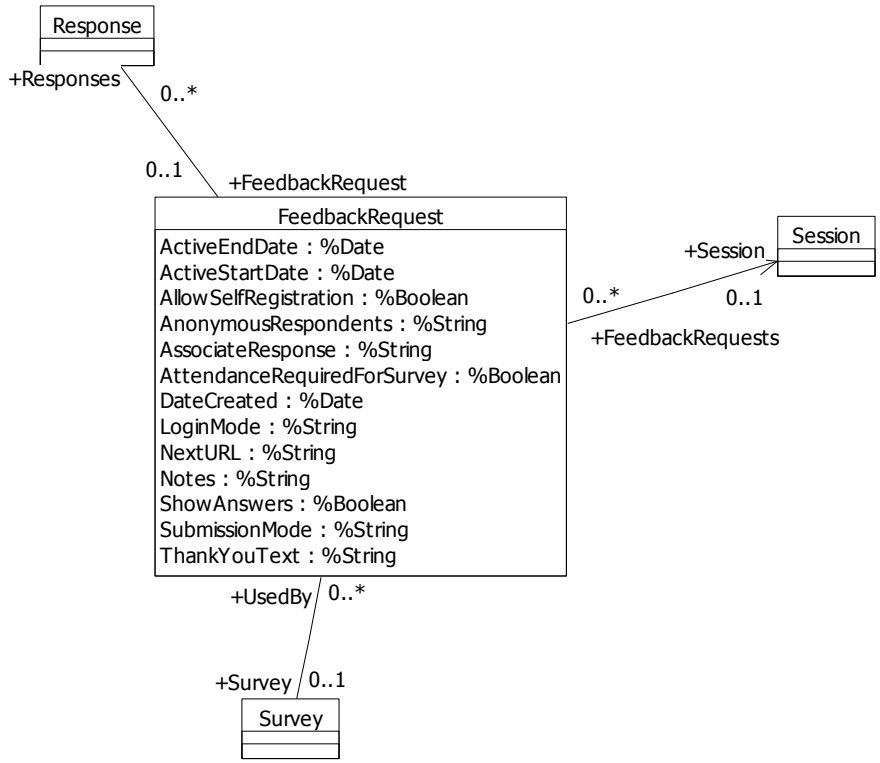


Figure 6-11: The FeebackRequest Class

The ID of the feedback request is the point of reference for a survey's use, and is used to build a unique URL that is used for survey access. The URL is created via the `SurveyURL()` method, which accepts the name of the server and outputs the complete link to access the survey. There are many different properties in the `FeedbackRequest` class, all of which are explained briefly below.

- `ActiveStartDate` – This is the date on which the survey can first be accessed for submission.
- `ActiveEndDate` – This is the date on which the survey can last be accessed for submission.
- `AllowSelfRegistration` – This Boolean field allows survey respondents to add themselves to the list of students registered for the course, and then take the survey.
- `AnonymousRespondents` – This is an array of type `%String` that is used to store the names of those respondents who performed a Verification Login and submitted their response anonymously (see `AssociateResponse`)
- `AssociateResponse` – This switch controls the way in which responses are associated with people who give them. There are some circumstances in which a record of respondents is needed, but it is not necessary to know who said what. This field allows the association between login and response to be controlled. It can take on one of three values:

    - `Yes` – This means all users are associated with their responses in the system.
    - `No` – Users are asked to perform a "Verification Login", which requires them to be registered in the associated course and provide their username and password, but then associates their response with the Anonymous profile. Their name is added to the `AnonymousRespondents` array following the survey submission. There is no way to connect the response with the respondent in this case.
    - `StudentChoice` – Respondents are given the choice between performing a "Verification Login", and a regular login.
- `AttendanceRequiredForSurvey`– Setting this Boolean property to true will block students from the survey if they have been marked as "Absent" in the session in which this `Survey` is used.
- `DateCreated` – This property is auto populated with the date upon the creation of a new `FeedbackRequest` object.
- `LoginMode` – This switch controls the way in which respondents identify themselves prior to taking the survey. It can take on one of three values:
    - `LoginRequired` – All respondents must be registered for the associated course, and provide a valid Username and Password in order to be brought to the survey.
    - `AnonymousOption` – Users can choose between logging in (which requires their being registered for the course and providing a Username and Password), and taking the survey anonymously.
    - `AlwaysAnonymous` – All users take the survey anonymously, and are brought directly to the survey page.
- `NextURL` – This property is a `%String`, which contains the web location to which the respondent is redirected following completion of a survey. If this property is blank, then no redirect occurs and the respondent sees text thanking them (see `ThankYouText`).
- `Notes` – This field is used to record notes on this use of a survey.
- `ShowAnswers` – This Boolean field will be implemented in the future, and will control whether or not the correct answers are shown to the respondent following submission. This functionality is not yet implemented.
- `SubmissionMode` – This switch controls the number of entries that a person can submit. It can take on one of three values:
    - `Single` – Each respondent (based on their login) is limited to submitting one response.
    - `Changeable` – Each respondent can only submit one response, but if they visit the survey again, they will be shown their previous response and be able to change the answers.[54]
    - `Multiple` – Each respondent may submit as many responses as they wish.

[54] This functionality is partially, but not fully implemented at the time of this writing.

- `ThankYouText` – This string is shown to the respondent upon the completion of a survey, provided the `NextURL` property has not been set.

These properties allow the use of a survey to be tailored to the specific needs of the instructor for a given course and session. They represent the "presentation" of the survey, separated from the "data" (the objects within the survey itself). This separation enables the reuse of the same survey across multiple contexts, courses, and security settings. This allows instructors to share their surveys with their colleagues for use in different subjects, and allows the responses to be aggregated on either the "use" level, or the "survey" level. In addition to these presentation settings, the paradigm of requiring a survey to be used within a course allows for additional use-specific access control over that survey.

6.4.2 Course-Specific Access Control

The placement of a survey within a course setting (specifically a `Session` object) means that the "use" of the survey has access to all of the properties of the `Session` and its parent `Course` object. These properties allow for further control of how a survey is accessed and behaves. Specifically, the `Course` object could have a number of `Student` objects registered for it, which are therefore related to the `Survey` object for that particular use. This translates into a list of potential respondents, which can be used to control access to the survey. For educational feedback collection, the course list is the logical access controller for survey responses. By maintaining a common student list for every survey used in a course (i.e. the course registration), the task of entering valid respondents is eliminated (this is a substantial benefit over most other general use survey tools that are currently on the market).

Beyond the course list, there is also an attendance list that is maintained for each session within a course, and this is another reference upon which the survey use can draw. During the evaluation of customer needs, the use case was presented in which surveys would be used as follow-ups to class discussions, and would be included as part of the students' participation grade. This being the case, it was necessary to be able to limit survey access to those students who participated in the class discussion. The result was the functional requirement of being able to track those students who were absent from a course session. With an absent list, it is then possible to block those students from a survey who missed the session. This is another benefit of tying a survey use to a session through a many-to-many associative class.

The above two properties of Session-Survey interaction highlight the ways in which the same survey can have different use properties based solely on the context in which it is used (i.e. placing a survey within a course automatically generated a list of eligible respondents).

6.4.3 Context-Awareness of Instructor Specific Questions

The last major benefit of the many-to-many associated `FeedbackRequest` class is that it ties a list of course instructors to the use of a survey. This is key, since often course feedback is tied in specifically to the performance of the course instructors.

Therefore the collection of such feedback raises issues of confidentiality of student responses that might potentially be embarrassing if seen by instructors other than the one on whom the feedback is based. However, the problem of limiting access of the responses to the course instructor is complicated by the fact that in higher education there is often more than one individual that has instructional responsibilities for a single course (e.g. lecturers, TA, etc). Therefore, partitioning the results so that each instructor can see only those answers intended for him or her is a very difficult challenge. In fact, apart from the arrangement implemented in OnFORME, no other solution to this issue (of allowing instructors viewing access of survey results while partitioning them based on private information) has been observed in the range of applications in this space.

The solution implemented by OnFORME is enabled by the reuse paradigm, and its separation of data from presentation information. When the `InstructorSpecific` Boolean flag is set for a question, then the survey is rendered in the browser so that its presentation is dependent upon the context within which it is being used. This context-responsiveness causes the following to happen:

1) The list of instructors teaching that course is retrieved from the database.
2) The instructor specific question is rendered once for each instructor on the list.
3) The text "Referring to <Instructor Name>:" is inserted prior to each instance of the question.
4) The collected responses are flagged as instructor specific, and the `Answer` objects point to the instructor to whom the question was referring.
5) All reporting interfaces show instructor specific answers only to the person about whom the question was answered (or managers or administrators within the system).

This process solves the problem of multiple instructors in a class. Questions relating to the instructor's performance will (if marked as instructor specific) be rendered once for each instructor, but each instructor will only be shown their own personal responses when they enter the system to view the responses.

Besides serving to partition responses in a way that protects confidentiality, the instructor-specific feature provides automatic customization of a generic survey. The same survey can be used in a multitude of courses, and those questions referring to the instructor (or instructors) of the course will be automatically populated with the instructor's name for every course in which they are used. This flexibility and adaptability based on context-awareness of reused surveys presents a major innovative benefit of the OnFORME system.[55]

6.5 Class Summary

The survey engine described in this chapter is the central core of the OnFORME system. The ability to create, manage, and post surveys, as well as the ability to store and

[55] The concept of instructor specific questions can be taken a step further with the implementation of different levels of instructors within a course, and the ability to dictate the level of instructor to which a question applies. For more information, see section 14.1.

navigate their responses in an intelligent way, is vital for the feedback process to occur online. Figure 6-12 shows all of the classes that are involved in this half of the OnFORME architecture, and how they relate to each other.

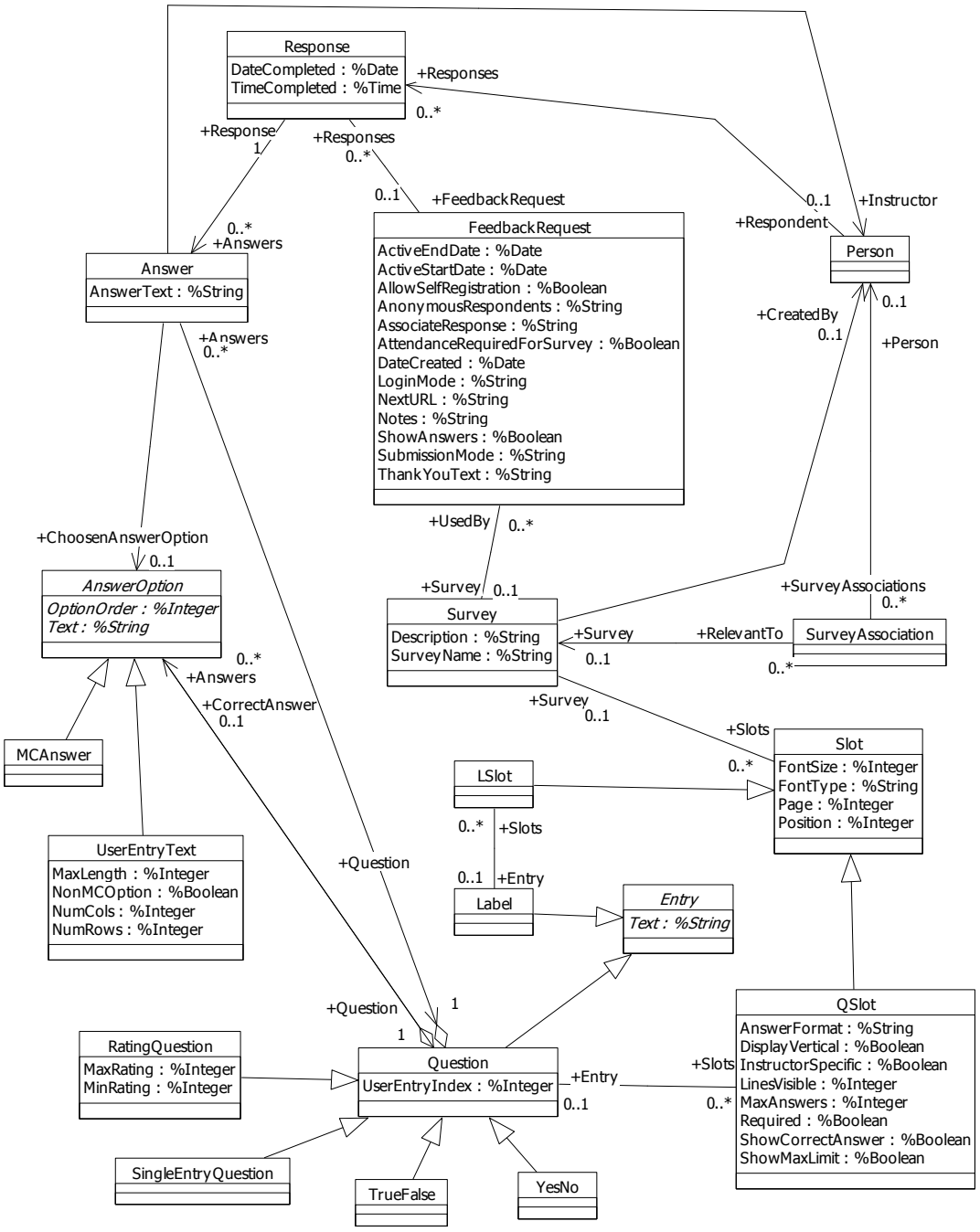


Figure 6-12: Interaction of all Classes used in Feedback Collection

7 User Interfaces

This chapter briefly discusses highlights of OnFORME's front-end user interfaces. It starts with a discussion of Caché Server Pages technology, which is the front-end platform used for OnFORME. It then proceeds to discuss the security paradigm and how the pages are partitioned by role. The conclusion discusses the various reporting features included in OnFORM.

7.1 Caché Server Pages

As a web application, OnFORME needed a web technology on which the front end of the system could be created. Caché includes a native web technology designed for building dynamic, scalable web interfaces to the Caché database. This technology is called Caché Server Pages (CSP). CSP technology enables web interfaces to be created programmatically, as HTML files with CSP tags, or as a combination of both.

CSP technology requires three components to function:

- Web Server – serves static content to client browser; handles HTTP requests[56]
- CSP Gateway – passes along CSP requests to the CSP Server
- CSP Server – handles the CSP requests, runs server-side code; passes static content back to web server to send to client.

The three components mentioned above work together to process any requests made by the client browser for CSP pages (`*.csp`), or CSP classes (`*.cls`). The web server handles all other web content.

In OnFORME, CSP pages contain several types of code:

- HTML – Industry standard markup tags used to control the appearance of static content in a web browser. HTML is rendered in the client's browser.
- CSP Tags – Caché specific tags that enable direct access to data and programmatic elements within the Cache Database. CSP Tags are evaluated within the CSP Server.
- DHTML – Industry standard script that is used to make content in a browser behave dynamically. DHTML controls content within the client browser.
- JavaScript – Industry standard scripting language used to dynamically control the behavior of a web page and web browser. JavaScript is executed within the client's browser.
- Caché Object Script – Caché programming language used to create methods and control behavior of a page before it is sent from the server. Object Script is executed on the CSP Server.

[56] In the OnFORME configuration at MIT and at InterSystems, the Web Server is Microsoft IIS running on Windows 2000.

- SQL – Industry Standard database query language that is used to pull data into the web page directly from the OnFORME database. SQL is executed on the CSP Server.

There are a total of 74 CSP pages included in OnFORME, which handle all respondent interaction, survey manipulation, course control and administrative functionalities. These pages are listed in detail in Appendix D, and a high level overview map of the system navigation is shown in Figure 7-1.

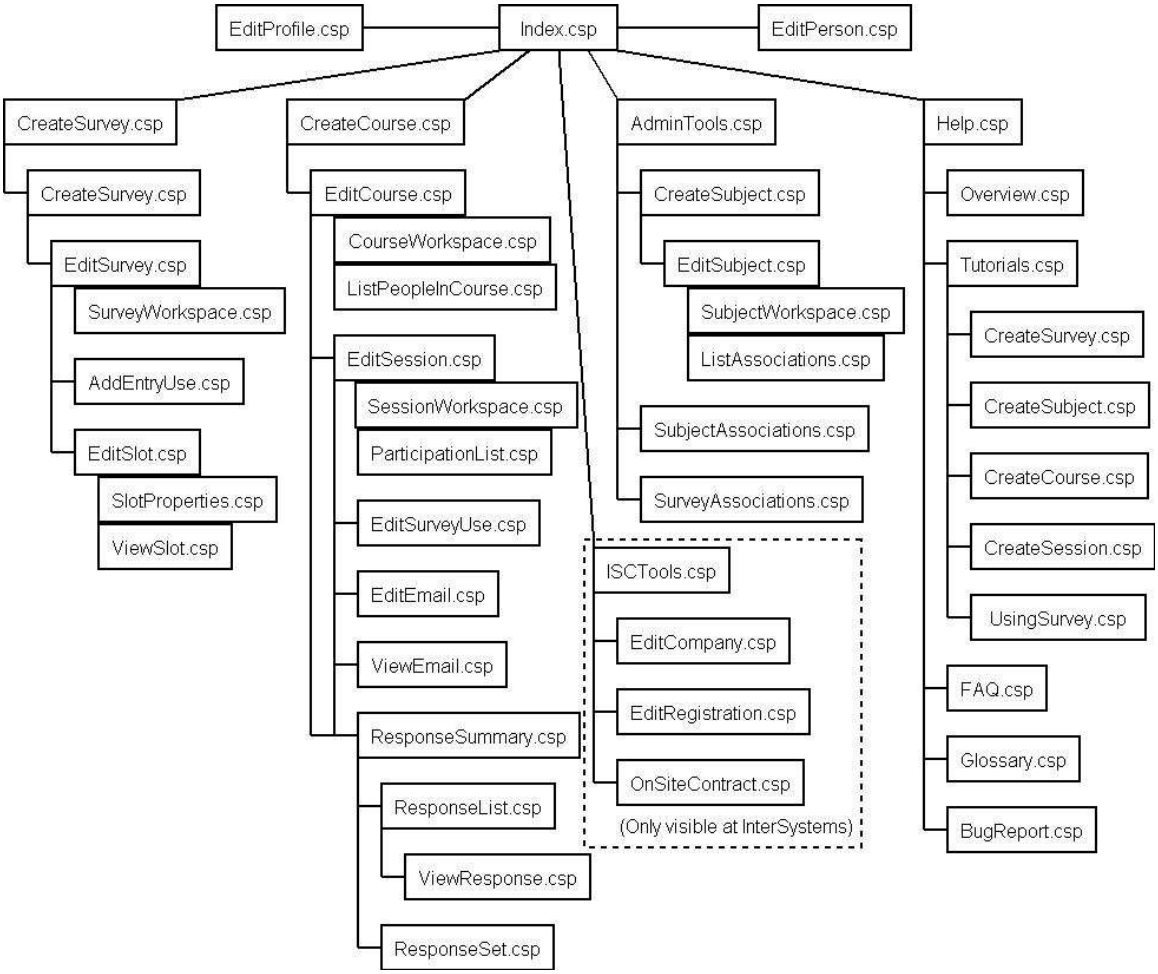


Figure 7-1: Diagram of OnFORME CSP Navigation

7.2 Security

There are two levels of security within OnFORME: authentication security and access security. Authentication security confirms that an individual is a known user within the OnFORME system prior to their being able to view any interface page within the system. Access security confirms that the user has the appropriate role to view a certain page, and it is handled on a page-by-page basis.

The security is handled in the `OnPreHTTP()` method, in which every CSP page calls prior to sending any content to the client browser. If either the authentication or access checks fail, then the requested content is never sent to the browser.

7.2.1 Authentication Security

The OnFORME system requires that for all pages except the login page, a valid username and password be recorded with the system prior to the page being sent to the client web browser. There are three ways in which the information can be retrieved:

- Manual Entry – When the user first comes to the system, unless they have previously stored their username and password as a cookie in their browser, they will be prompted with the `Login.csp` page to enter a valid username and password.
- Stored on Client Cookie – When a user logs in, they have the option of choosing to allow their login information to be stored on their computer as a cookie. If they did this at some point in the past, then when they go to access an OnFORME page, that cookie will be retrieved and the user will be passed onto the page.
- Stored in the Session – After the proper login information was either entered by the user or retrieved from a cookie, it is stored in the browser session and retrieved every time a page is accessed.[57]

When the client browser requests an OnFORME page, Caché checks whether or not a valid username and password are recorded. If validation occurs, then the page is sent to the client browser. If validation fails, then the login page is sent to the client browser, and following a successful validation on that page, they are redirected to the originally requested page.

7.2.2 Access Security

Following user authentication for a page request, OnFORME then confirms that this particular user has the role necessary to view this particular CSP page. Each role has an assigned Access Level, which controls what the user is allowed to see and do in OnFORME. The roles and their access levels are shown in Figure 7-2.

Role	Access Level
Student	1
Instructor	2
Author	2
Manager	4
Administrator	6
SuperUser	8

[57] It should be noted that this browser session is different from the `Session` class discussed elsewhere in this thesis. This session stores all of the information pertaining to that use of OnFORME. The information is stored in either a temporary cookie, or it is passed as a URL parameter from page to page.

Figure 7-2: Access Levels of the OnFORME User Roles

Each CSP page in OnFORME has a tag that lists the access level required to see that particular page. Figure 7-3 shows an access level definition for a page that only Administrators and SuperUsers can view.

```
<CSP:Parameter Name="AccessLevel" Value="6">
```

Figure 7-3: Example Access Level Parameter for Administrator CSP page

If a user requests a page whose Access Level is greater than that of the user, the user's browser is redirected to the `Forbidden.csp` page. Every CSP page that is part of the OnFORME application is set with an access level of "2" or greater. Therefore, all users of the system who have no role other than "Student" are unable to access any of the creation, formatting, or reporting interfaces of OnFORME.

7.3 Reporting on Survey Results

Currently, OnFORME is equipped with three ways of reporting data out of the system. Two of the reporting pages are presented in graphically pleasing formats so they can be printed (or cut and paste into a word processing document), while the third page is intended to present all of the information in a way that enables copying the data into a spreadsheet.

It is possible to access the reports from the Course Workspace (`CourseWorkspace.csp`), which is shown in Figure 7-4. The reports can also be accessed from the Edit Session page (`SessionWorkspace.csp`), which is shown in Figure 7-5. Both locations have a place next to each survey "use" that shows the number of responses, which is hyper linked to the Responses section of OnFORME. The links point to the Response Summary page and from there a user can jump to the other two pages.

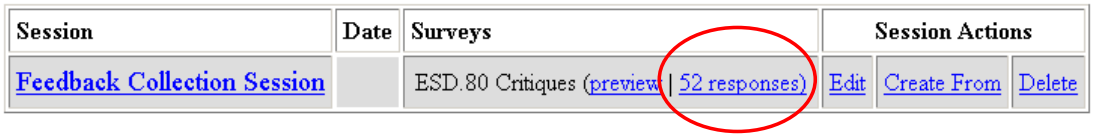


Figure 7-4: Accessing the reporting pages from the Course Workspace

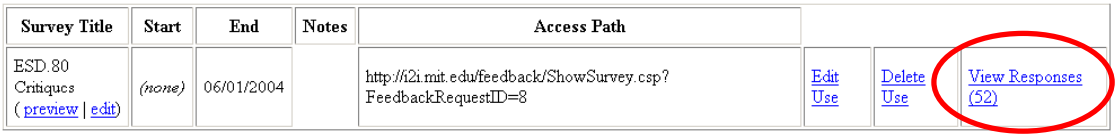


Figure 7-5: Accessing the reporting pages from the Edit Session page

7.3.1 Response Summary Page

The first results page is the Response Summary Page, contained in `ResponseSummary.csp`. The page displays an executive summary of all of the data collected for a `FeedbackRequest` object. For an example of this page, see Figure 1-23 in Chapter 1.

This page can be printed as a report, or cut and pasted into a word processor for further manipulation. If the survey contained any Instructor Specific questions, then this page would only show those questions that referred to the user logged into the system (unless the user is a manager or higher). The page also shows response statistics, and reports multiple-choice questions by percentages on the answer options.

From `ResponseSummary.csp`, the user can navigate to the other two report pages by way of the Change View dropdown box in the upper right-hand corner of the page (see Figure 7-6).

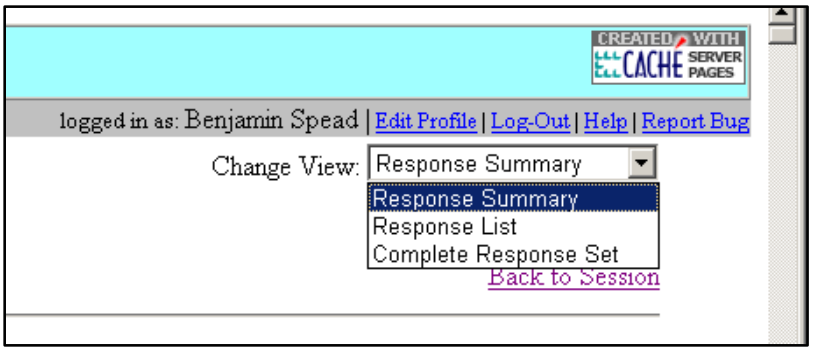


Figure 7-6: Navigating to the Other Response Pages

7.3.2 Response List Page

The Response List page is ResponseList.csp, and it shows a list of the responses separated by respondent. From this page, each response can be viewed individually in order to see a user's answers all within the same context. For an example of the Response List page, see Figure 1-19 in the example in Chapter 1. Additionally, the Response List page will show a list of all students registered for the Course who have not yet submitted a response (used for accountability purposes). Finally, there is a list of those users who responded using a Verification Login, which means that their response is one of those marked "Anonymous" in the response list.

The Response List page can be printed to create a record of the respondents and the times and dates of their responses. Additionally, each individual response (which opens in a pop-up window) can be printed or cut and pasted into another program.

7.3.3 Complete Result Set Page

The Complete Result Set page is `ResponseSet.csp`, and it is a data-dump of all of the question and response data for a single `FeedbackRequest` object in the database.

This page was not intended to be viewer friendly. Instead, it was created as a first attempt at providing a means to export the data into another program (namely a spreadsheet).

This page contains a single very large table that holds all of the data related to the `FeedbackRequest` object. The headers of the table contain the information from the questions in the survey. Each `Question` object text (in a column) is followed by a column for each `AnswerOption` object text. If the question has the option for a free text answer, then that column comes last. The exception to this rule is for Dropdown questions. Since they may have a large number of answer options, there is the same number of columns as there are allowed answers.

The rows of this page correspond to individual responses connected to the `FeedbackRequest` object. The name of the respondent is given, followed by the time and data of the response. Then, if the respondent chose a particular answer option, a "1" is placed in that column (except for Dropdown questions, which place the answer option text in the cell). If the option was not chosen, the column is left blank. This approach was used so that responses could quickly be tallied in spreadsheets, based on the Boolean data in the rows. Finally, any text entered by the user is also placed in the appropriate column.

The data export feature of OnFORME is very limited and is not considered to be complete. Currently, the way that data can be taken out of OnFORME is by selecting the entire table in `ResponseSet.csp`, copying it, and pasting it into a spreadsheet. This method is functional at present, but far from optimal. Future work will include adding the ability to export this table as a CSV (comma separated value) file.

7.3.4 Response Security

Each of the three response reporting pages has a limited level of built-in security to prevent unauthorized access. Besides the fact that only instructors or higher with valid logins can get to the pages (see previous section on security), `ResponseSummary.csp`, `ResponseList.csp`, and `ResponseSet.csp`, contain an additional level of protection in that their URLs cannot be accessed directly. These three pages are set to be "Private" pages, which means that their URL and its parameters are encoded and access to the page must be from a hyperlink elsewhere in OnFORME. The CSP tag specifying this behavior is shown in Figure 7-7.

```
<CSP:class private=1 encoded=1>
```

Figure 7-7: Creating Private Result Pages

The sole parameter passed into these three pages is the object ID of the `FeedbackRequest` object being reported. By making these private pages, users are prevented from typing random FeedbackRequestID values into their browser and being able to see survey results for classes with which they are not associated.

8 Development Methodology

Now that the discussion of the implemented architecture and data model has been completed, it is instructive to review the process used throughout the development process. Owing to the constraints places on the project (especially the lack of time), it was necessary to employ development methodologies that differ from those used for large structured software projects. This chapter reviews the methods used during the development of OnFORME, and places them within a methodology context.

8.1 Project Goals and Agile Approach

The high-level goals of the system, as set out by the initial customers and their most strategic use cases are listed below.

- Create an application that will allow for the creation of web-based feedback instruments (surveys, polls, etc).
- The application must allow for maximum feedback object reusability, in order to minimize the time required for instructors to populate the feedback instruments with content (questions, text, etc).
- The application must enable the organization of the use of feedback instruments within an educational context.
- The range of question functionality available for use within feedback instruments must be as flexible as is reasonably possible.
- The application must be fully stable and fully featured (i.e. a standalone finished product) by the end of the final development cycle (i.e. by the time I graduated).

In light of these goals, and an eight-month development window for a one-person team, it would have been very difficult to complete the project on-time while employing the structured software engineering approach of pre-specifying every piece of the architecture, defining every class interaction, and completing all documentation prior to the commencement of writing code. The project constraints didn't allow enough time for this approach, so there was little choice but to employ more agile and less rigid methods in an attempt to complete the project on time.

A lack of rigidity in a software development process does not equate to a lack of structure or strategy. In fact, there are many software development processes that focus more on agility than prescriptive rigor in an attempt to accomplish their goals in a shorter period of time. Prior to the post-development process evaluation of OnFORME, I had not been aware of any of these agile practices by name, but in taking a pragmatic approach to balancing my project requirements and constraints I unknowingly gravitated towards many of the central focuses of agile software development. In an attempt to congeal my strategies and priorities into communicable structure, this chapter presents a review of the processes that I followed while implementing OnFORME. This review is structured by comparing the design, development and deployment steps of OnFORME with those contained in the agile programming methodology, Extreme Programming (XP). There was not a decisive attempt to model XP through the life of this project. However, it is informative to match up design reality (which was in fact agile in nature) with the ideal

practices put forth by XP enthusiasts. For this comparative analysis, I used the structured 12 Practices of XP, as outlined by Maurer and Martel in their 2002 article describing XP in "IEEE Internet Computing" (Maurer and Martel, 2002, pp. 86-90).[58]

The ensuing discussion contains a mapping between my work and Maurer and Martel's "12 Practices" organization of Extreme Programming. The descriptions of these practices have all been abridged from Maurer and Martel's 2002 article. For each of the 12 practices I present the description (as summarized by Maurer and Martel), and then I discuss the steps I performed which fulfilled (or did not fulfill) the various practices. Again, it is important to note that during the development process, it was not purposefully attempt to implement an XP approach – I merely employed agile processes that, in my best judgment, provided me with the best chance of completing the project according to the design goals. Due to the constraints placed on the project, my approach was much less process driven than that of a formal software development project, and was more in the spirit of the "Manifesto for Agile Software Development", which emphasizes:

> Individuals and interactions over processes and tools
> Working software over comprehensive documentation
> Customer collaboration over contract negotiation
> Responding to change over following a plan (K. Beck et. al., 2001).

Looking at XP in general, Maurer and Martel categorize the process as 12 related practices, which emphasize small teams, simple functional code, automated test drivers, and refactoring.[59] The methodology employs pair programming and minimizes paper-based documentation (focusing on well-documented code as the replacement). It aims to be a very lightweight software design process, honed to be agile in order to respond to time to market constraints (which was a sizable factor during the development of OnFORME). However, the authors also emphasized that not all of the principles are appropriate in every situation, and it is not always possible to do all of them.

Prior to proceeding, it should be noted that over the last couple of decades, there have been many attempts to create different types of lightweight software design processes. At their conception, each of them was lauded as the breakthrough that would revolutionize software development, but after the hype died, ended up having less success in process penetration than hoped. One of the reasons for this is that agile programming contains downsides, which do not lend themselves to enabling industry-wide acceptance. For example, as some approaches focus on getting a product to market as quickly as possible, shortcuts are taken which could affect the long-term maintainability of the code base. While this is not a detriment in markets that turn over quickly, this side effect prevents agile methods from being embraced by projects that have long time horizons. It is entirely possible that the agile approach used for OnFORME could prove to cause

[58] It should be noted that there is more than one way to categorize and group the practices commonly associated with XP. For another example which breaks XP into 4 categories containing 28 Practices total, see Wells, "Extreme Programming: A gentle Introduction" (2003). Maurer and Martel mention the majority of principles listed by Wells, but they are bundled into 12 Practices for their analysis.
[59] Refactoring means rewriting working code to make it more simple or streamlined.

maintenance complications that arise in the future. However, this is a risk that had to be taken given how imperative it was to have a fully functional application prior to graduation. The history of graduate student software projects demonstrates that partially or mostly functional software has no chance of surviving following the developer's graduation when no one else has a vested interest in seeing the work completed. However, by finishing OnFORME and engaging customers prior to my graduation, there is at least some chance that the software will survive to be useful, even if the agile approach increased the challenge of future maintenance.

The following sections describe the individual practices, list the XP principles, and explain how those principles where (or were not) matched by the development efforts which I undertook during the development of the OnFORME application. Due to the size of data and examples being presented, the following sections are laid out in bulleted form.

8.2 Customer Satisfaction

The following XP Practices focus on improving customer satisfaction with the finished product, realizing that a perfectly coded application that does not align fully with the customer's needs can be a practical failure (Maurer and Martel, 2002, p.87).

8.2.1 On-Site Customer

Practice Details:

- It is recognized that all of the functionality requirements for a system are difficult to fully capture at the beginning of a project.
- Often, requirements change in dynamic web-based environments.
- At the onset, requirements are documented through User Stories (textual use-case descriptions).
- Whenever possible, an on-site customer representative works with the development team so that programmers can get customer input immediately, and they do not need to speculate about customer preference.
- Customers can therefore change requirements on short notice.

Practice Implementation with OnFORME:

- The list of customers (or users) for OnFORME grew incrementally as the project progressed. The first two customers represented both academic educational use and corporate educational use.
    - i.e. Dr. Joel Cutcher-Gershenfeld and Jim Breen.
- I was in very close proximity to the customers during the entire development cycle.
    - e.g. Direct contact with customers at MIT and InterSystems allowed for easy communication of requirements.
- When the customers came to better understand the scope of the application, they requested additional features.
    - e.g. After further considering how the reporting data might be used statistically, Dr. Joel Cutcher-Gershenfeld requested a change in how the data was presented so that it would be easier to import into his statistical analysis program (SPSS).

- Frequent interviews and usability tests were conducted among customers and potential customers of the system.
    - e.g. As part of my continual interfacing with the customers, I performed more than 30 customer interviews, a list of which can be found in Chapter 15: References.
- The functional requirements for OnFORME were created through understanding usage scenarios gleaned from interviews and scenario modeling (extrapolating knowledge of how educational environments work).
    - e.g. Prof. Bill Nuttall explained during one interview that as an instructor, he needed OnFORME to be able to produce executive summaries of the survey results, containing the data in a compiled, succinct format (which used graphics to help communicate the key points). His desire would be to have something that he could print and take along on the train to read while he traveled.

8.2.2 Small Releases

Practice Details:

- Keep release cycles short, and each release produces a useful software system that has increased value.
- Short cycles reduce customer risk, because things that don't work can be discovered more quickly.
- This process also helps developers deal with changing requirements, and reduces impact on planning errors.

Practice Implementation with OnFORME:

- It was easy to release updated versions of the software on a fast basis, due to the fact that I did not have to navigate changes with a common code base changed by other programmers. Therefore, after the changes were made to the code base and tested for stability, I would update the changes to the production servers at the first available opportunity.
    - e.g. After the initial MIT deployment of OnFORME on February 6th, the server was updated with 17 incremental releases before the end of March. The releases occurred on 2/7, 2/9, 2/11, 2/12, 2/14, 2/23, 2/25, 2/27, 3/2, 3/3, 3/6, 3/9, 3/17, 3/22, 3/23, 3/27, and 3/29.
- In fact, the quick update cycle allowed the incremental changes requested by the customers to be delivered in very short order (sometimes, the change would go into production that same day).
- There were also a couple of occasions where minor bugs were found in the system, and I was able to fix them right away and install an incremental release on the application server.

8.3 Software Quality

The following XP Practices focus on trying to insure the output of high quality, manageable code, without slowing down the development process (Maurer and Martel, 2002, p.87).

8.3.1 Metaphor

Practice Details:

- This is a picture that represents a coherent view of the end result system, by representing "what we are trying to do."

- This metaphor makes sense on both the business and technical sides, allowing all developers working on the project to have a unified vision of the expectations for the finished product.
- Sometimes it is a single user story that gives everyone the idea of the system basics.
- This can be a sort of high-level software architecture.

Practice Implementation with OnFORME:

- The purpose of the "Metaphor" is to communicate a unified vision of the technical and business goals of the project to all developers. As I was working independently, I did not have the same need to create a project "vision," because there were no other developers with whom to communicate.
- However, while a metaphor was not necessary to communicate the design goals internally, I did find it helpful to construct succinct goal statements that were used to communicate the vision to potential users (customers). These statements included:
    - "I am attempting to create a system which can made feedback maximally ubiquitous and maximally transparent within educational environments."
    - "Currently, when a teacher thinks 'broadcast communication,' they open their email client and accomplish the communication goal in a under a couple of minutes. Ideally, when they think 'structured request for feedback', they ought to be able to point a browser to OnFORME, and accomplish that goal with an equivalent investment of time and effort to that of email."

8.3.2 Testing

Practice Details:

- Automated regression testing is an important part of keeping all code functional.
- The customer defines functional (acceptance) tests implemented by the development team, which verify the business values that the system is meant to contain.
- A feature lacking an automated test is considered to not exist.
- Programmers write unit tests prior to writing the application code that will be verified by those tests.
- Test drivers act as detailed specification of the methods' functionality, by enforcing functional expectations in a way that paper documentation cannot.

Practice Implementation with OnFORME:

- The XP Testing practice is one in which I did not reach a comprehensive level of equivalence in my process. XP testing is exemplified by writing the unit tests prior to the functional code, and all unit tests run at all times. I made no attempt to reach this level of automated testing in my development method.
- However, I did implement one unique testing tool, which was entirely in the spirit of XP Testing (although its scope was narrower).
    - e.g. While working towards the Beta release of OnFORME, I made the mistake of only testing the application against a single browser (IE 6.0). My mistake became evident further into my process, when I discovered that Microsoft Internet Explorer (IE) is much more lenient than the other mainstream browsers, and therefore much of my coding that worked in IE did not work elsewhere. After I spent the time changing my interfaces to be cross-browser compatible, I wanted to make sure that I was not surprised in the future by a cross-browser incompatibility. Recognizing my tendency towards always opening the same browser, I decided that it was important to ensure that I was frequently switching between

browsers, so that incompatible changes which might slip through testing would not be able to remain undiscovered for long. I spent a good deal of time creating a DOS batch file that randomly opens one of my three browsers (IE, Mozilla or Netscape). By connecting this batch file to my web-browser hot-key, I could make sure that I was cycling through the different browsers at regular intervals, and code changes which "broke" functionality in one of the browsers could be found more quickly. While this is different from an XP "unit test," it does line up with the spirit of continual bug checking, as I was constantly using browsers in my development and testing processes.

8.3.3 Simple Design

Practice Details:

- XP Assumes that requirements are always changing, and that the cost of changing the code is not exponential.
- The focus is on solving today's problems, rather than trying to anticipate future challenges.
- The best designs run all test drivers, have no code redundancies, have fewest possible classes and methods, and are easy to understand
- XP does not invest in up front analysis and design, as it trades potential savings of anticipated change against that of wrongly guessing the system's future direction.
- Keeping the design simple allows the team to work together with no documentation outside of source code.

Practice Implementation with OnFORME:

- Like XP, the OnFORME development process assumed dynamic user requirements, and that it was not possible to understand all of the requirements in advance.
- Additionally, there was a focus on removing all code redundancies, to allow for easier maintenance of the code base, and code that was easier to understand.
- Also, especially early in the design cycle, the focus was on implementing the very basic minimal amount of functionality, in order to create a beta product as rapidly as possible. The focus allowed quicker feedback and usage observation from the beta version, which gave me a better idea of how to prioritize the remaining feature requests.
- One aspect of the development methodology did differ somewhat significantly from that of XP, which was the amount of preplanning which was done for OnFORME. Of the four months between project inception and Beta release, two of those months were devoted entirely to customer interviews, usage case formulation, architecture design, and class documentation. While this was not the most "agile" way to begin a project with such a tight schedule, the highest strategic priority was placed on perfecting the underlying data models used in the system. This focus was used because the back-end of the application was database driven, and therefore drastic changes in the data model would have resulted in the necessity of data migrations on the test and beta systems. While some recommend constant migration as the preferred agile approach (D. Wells, "XP and Databases"), I was not aware of this at the time. Had I been aware of it, I would have weighed the ramifications of the time required for migration, and it is likely that I would have still chosen to invest heavily in upfront design.

- By investing more initial effort in an attempt to analyze and map out comprehensive system architecture, my methodology was closer to those of Crystal or DSDM (which put more focus on preplanning), as opposed to that of XP (Boehm, 2002, p. 66).

8.3.4 Refactoring

Practice Details:

- The programmers refactor the base code, by changing structure to improve understandability and maintainability, without changing functionality.
- When a feature is complete, the programmer must ensure that existing structures are the simplest way to run all tests, or the code must be refactored.

Practice Implementation with OnFORME:

- Although I did not approach refactoring in a methodical way, the concept of code simplification was always a part of my development methodology.
- Often I found that as I climbed the learning curve for the technologies I had chosen to use for my application, I would acquire new methodologies and tricks that simplified various parts of my design. Where possible, I would return to earlier bits of code in which I had faced a similar challenge, and apply the new learning there as well. In so doing, I was constantly simplifying the code and improving its consistency.
    - e.g. Early in my implementation, I had approached the need for ordering person lists alphabetically with an algorithm that places all of the names in a list and then loops through the list, pulling out the names in alphabetic succession. I later decided that a cleaner way to accomplish the same result would be to create a class inheriting from the List class, in which new items added to the list were placed into it in alphabetical order. This cleaned up my other method code considerably, as the `AlphabeticalList` object merely needed to be populated with all of the names, and then the names would automatically be extracted in alphabetical order.
- Besides refactoring method implementation, I also worked to make sure that the object relationships were as simple as possible. When I found an improvement that could be made to enhance the readability and increase simplicity, I would take the take to change the object model and update all of the pieces of code impacted by that change.
    - e.g. My original data model used a `FeedbackRequest` class which had a many to one relationship with Surveys and a many to one relationship with Sessions. Several months into my implementation, I realized that my Response modeling had a parallel structure, in that each response points to a Survey and a Session as well. I decided that my object model would be much simpler if I replaced Response's relationships to `Survey` and `Session`, with a many to one relationship with `FeedbackRequesst`. Therefore I changed the object model, and all of the associated parts of my code that used that part of the model. Refactoring made future coding on that part of the model much simpler.

8.3.5 Pair Programming

Practice Details:

- All code is written by two people working on one machine, where:
    - One person controls the keyboard and mouse, pragmatically focusing on broad issues about whether the code will work and how it can be simplified.

    - The other thinks strategically and decides if this is the best way to implement the functionality.
- The pair switches roles throughout the day.

Practice Implementation with OnFORME:

- This is considered to be a central part of the XP methodology, but for obvious reasons pair programming was not used during the development of OnFORME.
- However, while I did not have two people working on the code, I was able to apply one of the principles behind the pair programming concept. An important aspect to pair programming is the value found in having separate thought processes dedicated to both pragmatic implementation, and strategic implementation. While I could not effectively do those two things in parallel (as is possible with two people), I did make a concerted attempt to do them serially. This was especially true during the days in which I was focusing on application development from dawn to dusk. On those days, I was not able to code during every minute (e.g. on the subway), so I would use the non-coding time to map out strategic approaches to upcoming problems (or think through refactoring those which I had already tackled but was not happy with).
    - e.g. During one refactoring phase in which I was working on the student attendance control mechanism, I woke up early to get some code written (pragmatic), which was followed by time in the shower rethinking the architecture (strategic), followed by a coding session on the commuter rail (pragmatic), followed by a T ride (strategic), and then a day coding (pragmatic). It was actually during the strategic times away from the code that I hit upon alternative coding structures that ended up working the best. Had I been actively coding during all of that time (without strategic periods), I would likely have not discovered the simpler data structure solutions (or it would have taken much longer to discover them).

8.4 Project Management

The following are the XP practices that are intended to reduce management overhead while keeping customer's interests at the forefront (Maurer and Martel, 2002, p.88).

8.4.1 The Planning Game

Practice Details:

- XP takes both business priorities and development team realities into account.

1) **Scope and Priority (Business Priority):**
    - Which features are most important?
    - What must be added now?
    - Which features can be postponed?
2) **Release Date (Business Priority):**
    - When must the next release be available?
3) **Estimates (Development Priority):**
    - How much effort is required to implement a new feature or fix a bug?
    - How much work can be put into the next release?
4) **Consequences (Development Priority):**
    - How will various business decisions impact the development process and development effort?
5) **Process (Development Priority):**

    - How will the team work together and be organized?
- Marketing or onsite customers represent the business interests.
- Effort is estimated using "ideal engineering time" (IET), which measures a task's difficulty or complexity
- XP also uses "velocity", which determines how many IET points a team can use within a certain time period (velocity is determined based on past experience).
- The goal is to match customer expectations with realities of the developers' expertise.

Practice Implementation with OnFORME:

- Arguably, the principles contained within the XP concept of the "Planning Game" were the central driving forces behind the development decisions I made while creating OnFORME. This was a bottom-up design approach from the outset, where the customer needs were the driving forces of functionality. Trying to juggle the customer requirements with the realities of the time, labor and capital constraints placed on the project was a daily focus.

1) Scope and Priority (Business Priority):

- One challenging aspect of designing and implementing the application was that multiple customers would use it. Therefore, while the different users were always closely aligned with and giving input into the development process, in the end I (as the developer and architect) had to weigh all of the requests and prioritize them strategically.
    - e.g. One of Jim Breen's requirements was that the application be able to track student registration payments. However, since this was not a need shared by the other customers, I decided to hold off on that requirement until I had released a production grade application to my academic customers. Their basic functionality would not be available as early if I focused on Jim's requirement, and meeting Jim's requirement would not do him any good unless all of the feedback functionality was complete as well.
- In order to organize the action items coming from different customers (both feature requests and bug fixes), I maintained a request-list that organized the action items by type, requestor, and priority (which I assigned to be a numerical value from 1 to 7).
    - e .g. To date, the "punch list" contains a total of 215 action items, and of those, the 115 with the highest priority have been completed.
- Action items were prioritized from a business perspective based on the following criteria:
    - Scope of need – how many of the customers needed this feature
    - Urgency of need – what was a reasonable timeline for rolling this out in order to meet the business needs
    - Visibility of Feature – was this a feature impacting the experience of the customer's users (i.e. the students taking the survey), or only the customer (i.e. the instructor). If it impacted a wider range of users who were less tolerant towards a lack of functionality, then the feature was given higher priority.
- Based on the above criteria, new and old action items were often reviewed, and their priorities were adjusted accordingly.

2) **Release Date (Business Priority):**
    - The development timeline for OnFORME contained two inflexible milestone dates, and one somewhat flexible milestone date:
        - MIT Beta Release - early October
            - This release was necessary for early usage testing
            - In order to align with the academic calendar, the Beta had to be available near the beginning of fall term
        - MIT Production Release - early February
            - Since usage data was needed in order for me to complete the TPP portion of my thesis, it was imperative that OnFORME be rolled out at the beginning of the Spring Term.
        - InterSystems Classroom Training Production Release – Earliest possible date following the MIT release (actual release date was mid-March)
            - This release required additional functionality and architecture extensions to accommodate use cases for the corporate training environment.
            - This development cycle required additional use case development, user interviews, and requirement specification.
            - The goal was to provide this release as quickly as possible following the MIT release, to allow for additional usage results to be gathered for the TPP part of my thesis.
    - Beyond the current production releases, the following uses have also been planned, but will not occur in time to provide usage data for my thesis:
        - CARET Trial Release – sometime in May
            - CARET (the Centre for Applied Research in Educational Technologies) at Cambridge University has decided to do a trial of OnFORME to support feedback collection for the CMI M.Phils.
            - They are currently discussing the licensing arrangements, and hope to have the system operation in time to use it to collect feedback for this current Cambridge University term.
        - InterSystems eLearning Production Release – sometime in July
            - InterSystems is planning on using OnFORME to collect evaluation data associated with their eLearning initiative. Currently, they run four webcasts a week, and they intend to use this system to manage the schedule, organize evaluations, and add an additional student interface that allows students to self-enroll for the eLearning offerings.
            - There is an additional integration step that needs to take place with their eLearning content system. The work that I need to do to enable this feature is scheduled for completion by the end of June.
        - InterSystems Online Tutorials Production Release – sometime in July

- InterSystems is also planning to use OnFORME to collect evaluation data associated with their self-paced online tutorials.
- This initiative will require integration of a different sort, as the content of the browser based tutorials will need to be altered slightly to include redirects to surveys.

- Aside from the major releases that drove the development timeline, the development effort was characterized by almost 20 minor releases, which incrementally added functionality to the customers (without having them wait for another major release). This was documented more fully in the "Small Releases" practice description above.

3) **Estimates (Development Priority):**
   - While effort estimates concerning feature requests had a definite impact on how I prioritized those requests, I did not use anything as structured as the XP components of IET or velocity.
   - My estimates of effort were entirely qualitative, based on some or all of the following:
     - What is the scope of the change – will it impact existing code that will need to be refactored?
     - Do I have a similar process implemented elsewhere which I could adapt to implement this change?
     - What technology will be used to implement this change, and do I need to learn new features of that technology in order to complete the task?
4) **Consequences (Development Priority):**
   - Each of the business decisions discussed above had definite impacts on the reality of implementing the requested features. Looming deadlines or mission-critical bugs sometimes pushed back other efforts by necessity.
     - e.g. Just prior to the MIT Production Release of OnFORME, the need for a comprehensive set of user documentation rose as a high priority feature required for the release. Therefore, I had to put off many other small non-critical features, in order to concentrate on being able to deploy the application with user documentation and tutorials. The other features were not included in the initial production release, but were added through minor updates that followed in rapid succession.
5) **Process (Development Priority):**
   - The focus of this practice in XP concerns how the team works together and is organized.
   - Being a one-person team, there were not too many different organizational options.
   - The organization on my team was that I functioned as architect, developer, interface designer, tester, marketer, trainer, debugger, etc…

8.4.2 Sustainable Development

Practice Details:

- No one can work 60-hour weeks consecutively without quality suffering.
- Pushing a team too hard results in burnout and high turnover.

Practice Implementation with OnFORME:

- This was an XP Practice that I attempted to incorporate into my development efforts, but unfortunately I was not entirely successful. Due to the serious constraints on the development timeline, coupled with other priorities (such as classes and exams), I was not able to avoid a number of sprint sessions required to meet a deadline.
    - e.g. I put in several consecutive 60+ hour weeks just before the Beta and Production releases, and I most certainly felt the adverse effects of burnout (and my coding efforts were not as effective).
- On the flip side, I did find that the necessity of devoting time to my classes helped me to feel fresher for coding. Even though in general I was tired due to the total number of hours, during the weeks in which I could only code for 10 or 15 hours I was amazed at what I was able to accomplish in that amount of time.

8.4.3 Collective Ownership

Practice Details:

- Team members collectively own the code base, and anyone can add value at any time to any part of the code.
- This approach succeeds because of the automated test suite, with which it is known right away of a change to the code base broke a previously function piece of the software.

Practice Implementation with OnFORME:

- This practice did not apply to my development efforts, due to the fact that as the only developer on the project, I collectively owned the entire code base by default.

8.4.4 Coding Standards

Practice Details:

- Coding standards makes code easier to understand and improves consistency.
- Standards should be easy to follow and adopted voluntarily by developers.

Practice Implementation with OnFORME:

- The task of having consistent code is certainly made easier when only one programming style is used in creating an application. However, even though I was the only programmer, I still took care to use consistent methodology in different parts of the application to make readability easier.
    - e.g. my web interface pages have a large number of server-side methods. To keep all of these similarly organized, I created a utility class that holds all server-side methods, and each of my web pages inherits from the utility class. This provides a single point of reference when adding or changing server-side methods.
- Since most of the languages used in OnFORME were new to me at the onset, I discovered better ways of tackling problems as I spent more time on the project. When I found a better way of handling a particular type of problem, I would often return to my previous attempts in other places, and refactor that code to make it consistent with my later solutions. The later attempts became the modeling standards to which I tried to conform earlier bits of code.

8.4.5 Continuous Integration

Practice Details:

- Integrate code as often as possible (at least once a day), so that there is always an executable code available with latest changes.
- This often is done on a dedicated machine with all of the changes on it.

Practice Implementation with OnFORME:

- Due to the size of the team of people working on the code (one), it was not necessary to maintain a separate machine with the "latest" build. There was one machine which housed the code, and that was my development machine. This allowed changes to be tested each time against the existing code base, and finished code changes were automatically part of the body of code ready for the next build.
- In order to make sure that there would always be executable code available, I would always do an export of the functioning code base, prior to starting a major refactoring effort, or a major change in the architecture.
    - e.g. Prior to reworking the entire survey authentication portion of my code to incorporate the option for anonymous surveys, I exported my working code, and labeled it as "<date> PreAnon Change" to allow for an easy rollback if I needed to go back to my executable build.

8.5 Development Methodology Conclusions

While the development efforts behind OnFORME were certainly agile in nature, they were by no means without structure. The methodology decisions were made in light of the business and technical needs, as well as the overarching goals for the project. This process may not have been perfect, and were it to be repeated, the learning from this last iteration could be well applied. However, the process was responsive to the needs of the customer, while being structured enough to be able to produce a high-quality, finished result that met the goals and use case requirements on schedule.

9 Competitive Analysis

This competitive analysis was performed in to order to highlight activity that is currently happening within the educational feedback software domain, and to highlight ways in which the OnFORME application demonstrates functional innovation in this space.

9.1 Explanation of Analysis Parameters

It is important to note that there are many dozens of feedback applications that exist, but most are not targeted towards the educational market (rather, most aim at marketing or polling applications). This competitive analysis excluded non-education focused applications for the following reasons:

1. The goal of OnFORME is ubiquitous feedback use in educational environments, which assumes a possible high volume of surveys to be created and managed. The applications targeted at the educational market provide a way to organize surveys around class divisions, class lists, etc, while those aimed at marketing applications do not. Owing to the fact that general use survey creation tools do not fulfill one of the basic design goals for OnFORME, they were excluded from the list.
2. Additionally, the more robust general use survey tools that are intended for corporate use are priced such that their daily use in educational institutions is usually not an option financially.

The complete comparison matrix is contained in Appendix E and is organized on two axes. The columns represent the various applications that were chosen for inclusion in this study. The rows represent the different points of comparison used to differentiate the various applications.

9.1.1 Educational Feedback Applications Included in the Analysis

Aside from OnFORME, seven other Educational Feedback applications were considered in this analysis. The seven applications that were considered are on the list for one of two reasons:

1. Some of these applications were encountered as part of investigations of the local competitive space, which occurred during the development of OnFORME. For the most part, they are applications that are currently used as solutions in the MIT community, or in partner institutions (QTools, Sloanspace, and Blackboard).
2. The remaining tools were discovered during a post-design competitive analysis, and they were identified as the primary solutions available to institutions to handle feedback[60]. They were found through web searches,

[60] Additionally, there are other proprietary systems employed at some schools, e.g the UM.Lessons system which was built at and is being used by the University of Michigan. However, unless these systems have been made available to other institutions (e.g. Washington State University and Flashlight Online), they are not options worth considering in this analysis owing to the fact that they are cannot be adopted by a school looking to put a system into place.

educational assessment resource portals on the web, or by interview recommendation (Navigo, FAST, Quizlab, and Flashlight Online).

Additionally, the competitive search turned up one other educational assessment tool, namely WebCT, which is a Course Management System (CMS) that contains a survey toolset. However, WebCT was not included in this comparison due to difficulties encountered trying to collect information pertaining to its functional offerings.

Therefore, the final comparison included OnFORME plus seven other applications:

- QTools (MIT)
- Navigo (Indiana University and Stanford University)
- Blackboard (commercial CMS)
- Sloanspace (MIT)
- FAST (Mount Royal College)
- QuizLab (Person Education Inc.).
- Flashlight Online (Washington State University / TLT Group)

The completeness of this list (plus WebCT) was verified by a researcher at the Michigan State University, who did a similar point-by-point comparison while considering options for their upgrade path for feedback functionality (R. Espinos, April 9th, 2004). Therefore, this sample represents a best estimation of the educational feedback tools that could be adopted by members of the MIT community.[61]

9.1.2 Application Features Used as Points of Comparison

The points of comparison (listed in the left-most column of the Appendix E tables) were chosen for one of the following reasons:

1. They represent the background information helpful in understanding the scope of each application (these were collected for qualitative comparison purposes, and not used in the quantitative analysis)
2. They represent informative aspects of the application's underlying data model (these were collected for qualitative comparison purposes, and not used in the quantitative analysis)
3. They represent the use cases most important to the customers of OnFORME, and therefore they represent the functionality that exists in OnFORME.
4. They represent other features that have often come up during the development of OnFORME, although not as requirements (these where collected for informative purposes to determine to prevalence of such features among available applications)

[61] Late in the research process, it was brought to the author's attention that the company Question Mark also produces a product that is used for summative evaluations in educational environments. However, due to this information being discovered rather late, no attempt was made to add it to the competitive analysis.

There were 75 points of comparison (63 of which are discrete answer) chosen for this study. These points were broken down into nine different categories as shown in Figure 9-1.

Application Background:
Created By
System Provider Web Address
Contact Person
Contact Information
Intended User Base
Total Length of Development Cycle
# of People on Development Team
Initial Startup Cost (License)
Annual Cost (License)
Authentication Model
User Roles

Application Attributes:
Educational Environment Data Model
Direct Student URL Access
Local Hosting Option
Integrated User Database including all User Types
Student Registrations for Course
Session Based Absenteeism Control
Instructor Registrations for Course

Data Organization and Access:
Survey Reusability
Survey Reuse Presentation Parameters
Survey Cloning
Question Reusability
Question Reuse Presentation Parameters
Question Cloning
Author Based Survey Deletion Rights
Login Based Survey Viewing
Login Based Subject Viewing

Respondent Access Control:
Single Submission Control
Editable Submission
Multiple Submission Option
Respondent Login
Anonymous Responses
Anonymous Option
Verification Login
Verification Option
Student Self-Registration Option
Attendance Based Survey Access
Active Survey Period

Question Types:	Question Attributes:
Radio Question Type	Answer Alignment Option (V/H)
Radio with Textbox Option Question Type	Textbox Height Control
Radio with Comment Question Type	Textbox Size Control
Checkbox (Multi-Select) Questions	Font Control
Checkbox with Textbox Option Question Type	Required Option
Checkbox with Comment Question Type	Max Answer Limit (multiselect)
Dropdown Question	Max Answer Limit Display Option
Dropdown with Textbox Option Question Type	Question Grading (with "Right" answers)
Dropdown with Comment Question Type	Option to Show Respondent "Right" answers
Multi-Select Dropdown Question Type	Conditional Question Branching
Free Text Question Type	Instructor Specific Option
Ranking Question	Feedback Channel to Material Authors

Survey Presentation:	Communication Tools:
Custom HTML Wrapper for Respondent Pages	Email Functionality
Label Objects	Email Queuing
Customizable Thank You	Registration-Based Emails
Post Survey Redirect	Attendance-Based Emails
Multi-page Surveys	Rule-Triggered Emails

Other:
Cross-Browser Compatibility
Does Not Require Client-Side JRE
Question Limit
Additional Noteworthy Features
Additional Comments

Figure 9-1: Points of Data Collection and Comparison for Competitive Analysis

It is key to note that the list of features and functionality in this analysis is not meant to be a comprehensive list of functionality that exists in all applications in this space. That would be an interesting study, but it is different from the purposes of this comparison. The purpose of this comparison is to try to understand to what extent OnFORME implements functionality that is not widely seen in other educational feedback tools. Therefore, the majority of the points of comparison are taken from OnFORME's feature set, in order to determine the extent to which that feature set represents common practice, and to what extent it represents new functionality in this application market.

9.2 Background on Applications

The following gives a brief background on each of the applications included in this analysis. The highlights and interesting aspects of their approach are listed, giving some insight into the scope of that work.

9.2.1 QTools

QTools (or 'Quiz Tools') is the creation of the Academic Media Production Services, which is a cost-recovery services group at MIT. QTools was originally designed for a customer who wanted a way to quickly draft surveys for use over the Internet. QTools has gone through two design versions, where the first was considered a failure, and a client funded the second (which was launched as a successful product). The development effort is estimated to have been a yearlong process at 25% effort by a team of four (a project manager, a GUI designer, a graphic designer and a programmer). This equates to 12 months of fulltime effort for one person. The product is now available to members of the MIT community for instance licensing or on a subscription basis.

Data was collected on QTools via a 30-minute interview with the Project Manager Mark Brown, through observing a 60-minute QTools presentation, and through experimentation with their interface.

9.2.2 Navigo

Navigo was perhaps the most fascinating case considered in this analysis. Navigo was conceived as a joint research effort that began early in 2003 by Indiana University, the University of Michigan, and Stanford University. The vision was to create an open source general use assessment engine that could be used as part of an OKI[62] compliant architecture. A couple of months into the pre-project planning phase, the alliance broke up as the University of Michigan's contingent ran out of funds, and Stanford splintered off to work on its own efforts. Indiana University continued the efforts, and from April 2003 through November 2003, they had a fulltime development team devoted to the project. This team consisted of 1 project manager, 3 user interface designers, and 6 developers. Then in November 2003, the IU team merged their Navigo efforts with the work being done by the Stanford team (5 developers), and Stanford was officially part of the project again. By February 2004, they had begun implementing new functionality, and Stanford added an additional developer. Version 1.0 of Navigo is scheduled for limited release in May 2004. At that point (assuming that the team stays the same size), the project will represent 181 months of fulltime effort (not including work done by Stanford prior to the merged code base). The full open-source release of Navigo is scheduled for July 2005, as part of the Sakai 2.0 offering[63].

It is important to note that while Navigo can be used as one component in the Sakai (or OKI) environment, it will also have the functionality to act as a standalone feedback collection application. It is that functionality (as specified currently in the project plan) which was used for the basis of this analysis. Additionally, there is functionality that is intended for eventual use, but is not currently available, and may not be ready by May. In the comparison, this functionality is flagged as "In Process," and counted as a "Yes" for the competitive analysis. Also, it is noteworthy that the initial use cases on which Navigo focused were those involving graded assessments (although the intent is for it to be used eventually for surveys and polls as well), while the initial use cases of OnFORME were those emphasizing educational surveys and polls, and graded assessment functionality is in the architecture and can be later implemented.

Data was collected concerning Navigo via two 60-minute telephone interviews with the Navigo Project Manager, Lance Speelmon, who is the V.P. for Information Technology at Indiana University.

9.2.3 Blackboard Survey Manager

Blackboard Survey Manager is part of a larger Course Management System, which is intended to enable instructors to organize and manage all of their classroom activities. The version used for comparison was Blackboard 6.0, as that is the version used at Cambridge University, where Prof. Bill Nuttall provided system access so that Blackboard could be included in this analysis. Unlike other tools considered in this analysis (OnFORME, QTools, Navigo, and FAST), Blackboard does not provide direct URL access to surveys, which makes it difficult for it to be used more generally in

[62] OKI (Open Knowledge Initiative) is a cross-university effort to create an open standard for educational software interoperability, and open source applications based on that standard.
[63] Sakia 2.0 is considered by some to be the next generation of OKI.

educational environments. Rather, the surveys can only be accessed after students enter the workspace for a class (which requires their being registered for that class). Therefore anonymous submissions are not an option.

During the span of this investigation, it was not possible to discover the total amount of development time that the Blackboard Survey Manager tools required. The data used in the analysis pertaining to Blackboard was obtained through exploring the options available to an instructor through the Survey Manager portal.

9.2.4 Sloanspace Surveys

Sloanspace was created for the Sloan School of Management by a private software contractor. Similar to Blackboard, it is intended to be a comprehensive course management system, and there is no direct access to the survey functionality for the respondents. Due to the inability to create anonymous surveys, the types of feedback that can be collected by the system are limited to those to which the students are willing to attach their names.

All data collected on the survey functionality of Sloanspace was acquired through usage observation, and therefore no information is included in this analysis concerning the length of the development cycle, nor the size of the development team.

9.2.5 FAST

The Free Assessment Summary Tool (FAST) was created by Dr. Bruce Ravelli at Mount Royal College. The intent behind FAST was to create a tool that enabled instructors to request frequent anonymous feedback from their students for the purpose of improving their pedagogical methods. It has been an ongoing project for the last five years, and it has involved his own efforts and those if a single application developer (Z. Patz). They estimated that all of their efforts over the last five years would add up to 8-12 months of fulltime work for their team of two (18 to 24 months total effort for one person). FAST is offered to educators for free use on the Mount Royal College server, or alternatively, the source code can be purchased for $5000 and the application can be set up on an instructor's local server (they must also purchase a Cold Fusion license, and upgrades to the FAST software cost $1000). While it is intended for use in educational environments, FAST is rather limited in that it allows only 1 survey to be used for each course. If an instructor wishes to ask different questions, he must change the questions on that one survey, or make a pseudo-course to hold an additional survey. Also, this system was designed entirely around the concept of anonymous feedback collection, and therefore there is no way to create a validated list of respondents.

Data for the analysis on FAST was drawn from email conversations with Z. Patz, and through usage tests using the online FAST system.

9.2.6 QuizLab

Quizlab is an application that has been developed by Teacher Vision, a department of Pearson Education. Pearson developer, Vince Krist, created the current version over the span of 2.5 months. It is a package aimed at creating assessments that are used within

educational settings. The business model for Quizlab is that Pearson serves as an Application Service Provider (ASP), and all users keep their data on the corporate QuizLab servers. QuizLab is integrated with MyGradeBook, which automatically records the grades of graded online assessments (quizzes, tests), which is considered to be a major benefit of the system.

The comparison information concerning Quizlab was gleaned through an interview with the application architect (Vince Krist), and exploration of the online interfaces of QuizLab through a demonstration account.

9.2.7 Flashlight Online

Flashlight Online was a project that began at Washington State University, which acts as a host for this feedback service (there is no option to obtain the source code). The right to commercially license the application was sold to the TLT Group, which acts as a consulting body in finding users, and in training them how to leverage the benefits of the program. Client institutions buy an annual subscription to use the Flashlight Online system, for an annual cost running from $2500 to $5000. The novelty of Flashlight Online is that they have a peer review system for surveys, and a survey template and question test bank that includes over 500 peer review assessment questions, indexed by topic, target audience, and learning style. Another interesting feature of Flashlight Online is that it allows the survey authors to have full control over all of the HTML in their surveys. This allows advanced users to be able to add their own functionality that may not be native to the general application.

Information pertaining to Flashlight Online was collected through an interview with Steven Saltzberg (Senior Consultant of the TLT Group), an interview with Tom Henderson (Assessment Coordinator for the Center of Teaching, Learning, and Technology at Washington State University), and through experimental use of the online system.

9.3 Findings of Quantitative Analysis

The quantitative analysis was completed on the 63 features chosen for comparison. These features are each a discrete aspect of application functionality, which is assumed to reflect the flexibility of the underlying data models of the respective systems. Since the aim was the examination of the models, the choice was made not to compare the system features relating to the reporting of assessment results. This is because the creation of reports of any kind from the contents of a database is a fairly trivial matter, assuming that the proper data (e.g. the question responses) is stored in a manner that enables easy retrieval. Therefore, the 63 features are intended to highlight discrete functional capabilities of the application architecture, non-inclusive of reporting capabilities.

Of the 63 features, two of these were not designed into the OnFORME architecture. Those two features were:

- Ranking Type Questions
- Conditional Question Branching

"Conditional Question Branching" refers to intelligent assessments that dynamically change question order. "Ranking Question" refers to question types that are a series of questions that enforce uniqueness of an integer rank response across the series (e.g. please rank these 4 movies from 1st to 4th favorite). These features were included as data points in the qualitative part of the competitive analysis as they are found in a variety of general use survey tools, and therefore it was interesting to observe how prevalent they were among educational feedback applications. It is informative to note that none of the included systems contained Conditional Question Branching, and only one of them contained a Ranking Type Question (in that case, it merely functioned as a normal question, as the application did not seem to enforce the choice of unique ranking values). Due to the fact that these two functions did not exist in OnFORME, they were excluded from the quantitative findings (which focused on the prevalence of OnFORME functionality in the other applications).

For the purposes of binary feature comparison, those features that were designed into the underlying architecture (due to use case prioritization) were considered to exist, even if they were not yet complete. This was the case with certain Navigo and OnFORME features. This decision was made since the purpose of the exercise is to understand the flexibility of the underlying architectures and data models of the applications compared (even if that architecture is not yet fully implemented).

Removing the two features mentioned above that are not present in OnFORME, 61 features were left on which to conduct quantitative analysis across the various applications. This was done by identifying how many of the comparison systems contained each of the 61 features included for analysis. Within the feature descriptions (and elsewhere in this document), the term "survey" is used to signify a general feedback instrument, and is meant to signify any sort of assessment object which can be manipulated by the various applications (quiz, poll, assessment, survey, etc). The following sections examine the various levels of functional penetration.

9.3.1 OnFORME Features Common to all Systems

There were 6 features that all seven of the comparison applications shared with OnFORME. These features are listed in Figure 9-2.

Survey Cloning
Single Submission Control
Radio Question Type
Free Text Question Type
Cross-Browser Compatibility
Does Not Require Client-Side JRE

Figure 9-2: OnFORME Features Common to all Systems

These are the functionalities that can doubtless be called "common practice", and it is interesting to note that only 6 out of 61 features (or 10%) included in this study were common across all applications. A brief description and comment on each feature is listed below.

- Survey Cloning – This is the most basic way to allow survey (or assessment) content to be recycled within a feedback instrument database. It stands to reason that all applications would provide some means of copying content already in the system, thus preventing the necessity of re-keying the questions and answers.
- Single Submission Control – Some means of holding a respondent to a single survey submission is contained in the application. As a way to prevent people from "stuffing the ballot box", this is a logical feature for systems seeing widespread adoption.
- Radio Question Type – The most basic form of multiple choice question, a Radio Question has a series of possible answers with radio buttons, and the web browser enforces that only one answer can be selected.
- Free Text Question Type – The most basic type of freeform survey question, the Free Text Question type includes a text box into which the respondents can type their answer.
- Cross-Browser Compatibility – For a web application, the ability to work across a variety of different browser types is imperative.
- Does Not Require Client-Side JRE – JRE is the Java Runtime Engine, and is required for a client web browser to run Java applets used in a web application. OnFORME's administrative interface is quite complicated, and currently uses a Java application (the CSPBroker) for some of its functional compatibility. Owing to the challenge of a number of different JREs in general circulation, some of which have slight differences in compatibility, the OnFORME Java functionality in OnFORME is scheduled for removal. This point of comparison was included to see if other applications used Java, and they did not[64].

These features that are common to all represent the baseline functionality (or "table stakes") required for an online feedback system. They include:

1. Content reuse
2. Basic submission control
3. Multiple choice questions
4. Free text questions
5. Universal operability

Based on the prevalence of these features, it is safe to say that any new feedback systems aimed at this market would need this functionality at a minimum.

9.3.2 OnFORME Features Found in 6 of 7 (86%) of the Systems

There were 4 OnFORME features that were contained in exactly six of the seven comparison applications. These features are shown in Figure 9-3.

[64] It should be noted that Navigo uses the JRE for questions which require an answer in an audio recorded format, but as this is not a need that every user will have, that Java requirement was not included for the purposes of this analysis.

Educational Environment Data Model
Question Cloning
Respondent Login
Email Functionality

Figure 9-3: OnFORME Features found in 6 of 7 comparison systems

These features still fall within the range of "common practice", despite the fact that each of them is missing from one of the comparison applications. These 4 features represent 7% of the OnFORME comparison feature-set.

- Educational Environment Data Model – As this analysis is focusing on educational feedback tools, it is to be expected that most of them will have data models and organizational structures to support their use in educational environments. The only application that did not have some sort of an educational data model was Flashlight Online, which allowed for survey organization by "Group" rather than subject, class, or course (like the others). The assumption is that the "Groups" which are defined by the user will have some sort of mapping to their educational organization.
- Question Cloning – This is a content reuse option on a finer granularity than Survey Cloning. Almost all of the applications allowed individual questions to be copied for use in other surveys. The exception to this was Flashlight Online, in which entire surveys could be cloned, and then unwanted questions could be removed, but it was not able to clone questions from different surveys and compile them in a new survey.
- Respondent Login – Provides some means of user identification and verification prior to accessing the survey. This allows the survey respondents to be identified if the instructor so desires. The only application in which this was not present was FAST, since the application focuses solely on anonymous surveys, and therefore user identification is not necessary.
- Email Functionality – Basic communication capability native to the feedback application was adopted by almost every application (with the exception of Flashlight Online).

The features in 6 of the 7 comparison applications add the following functionality to the basic features of the previous section:

1. Data organization modeled on Educational Environments
2. Content reuse of a smaller granularity
3. Respondent identification
4. Basic communication

This set of features also represents a common base of functionality, which applications in this space can be assumed to require.

9.3.3 OnFORME Features Found in 5 of 7 (71%) of the Systems

There were 4 OnFORME features that were observed in exactly five of the seven (71%) of the comparison applications. These are listed in Figure 9-4.

Author Based Survey Deletion Rights
Login Based Subject Viewing
Multiple Submission Option
Checkbox (Multi-Select) Questions

Figure 9-4: OnFORME Features found in 5 of 7 comparison systems

These features represent majority (although not necessarily common) practice, as they are in roughly three quarters of the applications aimed at this market. These features are described in detail below.

- Author Based Survey Deletion Rights – The right of deletion of shared surveys is retained for the original author of the survey. This is not seen in applications that use group logins to share surveys rather than individual logins.[65]
- Login Based Subject Viewing – Areas of survey use can be dynamically made visible to different users at different times. This feature is helpful in allowing content to be shared across courses and for users to have access to more than one course.
- Multiple Submission Option – As a one step advance over Single Submission, this feature enables survey publishers to allow respondents to submit more than one response. This feature is useful for "rolling" assessments, in which a student may have feedback for the instructor more than once.
- Checkbox (Multi-Select) Questions – This type of question uses checkboxes in the web browser to allow a respondent to choose more than one answer to a question. This would be used in "choose all that apply" type scenarios. Together with Radio Button Questions and Free Text Questions, this completes the range of basic types of responses (single select, multi-select, and free form).

It was a surprise that Checkbox questions only appeared in 71% of the applications. It is likely that this is due to data models that are not flexible enough to handle more than a single response to each question (because in essence, choosing more than 1 checkbox is submitting more than one response to a single question). Additionally, the inability of users to be able to see more than one subject in the system may point to an architectural limitation.[66]

[65] A group login is a single login common to a cohort of people who share a common space. With group logins, the work of any single person in a group can be changed (or deleted) by others in that group. This model does not allow a survey author to maintain autonomy in their work.

[66] Alternatively, this perceived limitation could be the result of a strategic design decision, but unfortunately, it is difficult to conclude the reason without having access to the underlying data model.

9.3.4 OnFORME Features Found in 4 of 7 (57%) of the Systems

The next block of features begin to highlight areas of differential functionality between applications, as they are contained in just over 50% of the comparison applications (4 of 7). These features are listed in Figure 9-5.

Direct Student URL Access
Local Hosting Option
Instructor Registrations for Course
Anonymous Responses
Active Survey Period

Figure 9-5: OnFORME Features found in 4 of 7 comparison systems

The descriptions of the features are explained in detail in the following section.

- Direct Student URL Access – In order to allow easy access to published feedback objects, a URL pointing directly to the survey page is required. Without direct URL access, students must go through a series of navigational steps in order to access the survey. Applications that focus on total course management, such as Sloanspace, Blackboard and (to some extent) QuizLab, do not offer direct access to surveys for the students.
- Local Hosting Option – Without the option of maintaining one's own feedback application on a local server, an institution must be willing to store all of their course, user and feedback data on the server of the application host. These Application Service Providers (namely QTools, QuizLab and Flashlight Online) do not have the option for institutions to host their own feedback application.
- Instructor Registrations for Course – Often more than one instructor may teach a certain course, and in that scenario it would be important for the courses modeled in the feedback applications to have the ability to register more than one instructor for any given course, so that each of the instructors can inherit viewing rights to the survey responses. In actuality, only three of the applications give an easy means of accomplishing this. Additionally, one application (Flashlight Online) allows for a limited version of this feature.[67]
- Anonymous Responses – Many types of feedback have higher rates of response when the responses are collected anonymously. Therefore, for flexible feedback options, ability to allow respondents anonymity is very important. Those applications that embed their survey access within CMSs (Blackboard, Sloanspace and QuizLab), do not have the ability to allow the responses to be collected anonymously.
- Active Survey Period – The ability of setting start and end dates on surveys allows instructors to set up feedback objects in advance of use and control their access for some future date.

These features serve as functional differentiators between the applications included in this study, due to their inclusion in 57% of the sample set.

[67] Flashlight allows customizable variables to be used with surveys, and they recommend that if multiple professors are to be used with a survey, that it be codified in these option variables.

9.3.5 OnFORME Features Found in 3 of 7 (43%) of the Systems

There were eight of OnFORME's features (or 13% of the feature set) that were seen in exactly three of the seven comparison applications (43%). This feature set, contained in a minority of the sample applications, is described in Figure 9-6.

Integrated User Database including all User Types
Student Registrations for Course
Login Based Survey Viewing
Editable Submission
Customizable Thank You
Post Survey Redirect
Multi-page Surveys
Registration-Based Emails

Figure 9-6: OnFORME features found in 3 of 7 comparison systems

These features, like those in the last section, serve as differentiators (as they are contained in just under half of the comparison applications). They are described below.

- Integrated User Database including all User Types – Allows users of different types to be stored persistently within the system. Those applications that lack this feature force professors to create survey lists (and logins) each time a survey is published.
- Student Registrations for Course – Provides a means by which a class list can be created for a course, such that a group of students is maintained together, eliminating the hassle of creating lists for access rights, emails, etc. multiple times when different surveys are going to the same people.
- Login Based Survey Viewing – Surveys can be shared between different system users. This feature enables collaboration between assessment authors.
- Editable Submission – Allows those posting assessments (e.g. instructors) to enable the respondents to be able to return during a given period of time and edit their response.[68]
- Customizable Thank You – Allows assessment authors to choose thank you text that is shown to the respondent at the conclusion of the survey.
- Post Survey Redirect – Allows assessment authors to send respondents to a page of their choice following the submission of a survey.
- Multi-page Surveys – Allows survey contents to be spread across several pages.[69]
- Registration-Based Emails – Allows email to be automatically sent to all students that are recorded in the feedback application as registered for a particular course.

It is interesting to note that the number of features found in exactly three of the comparison applications is larger than any of the previous collections of common

[68]This feature is partially built into the data model of OnFORME, but the functionality has not yet been fully implemented in the user interface.

[69] This feature is partially built into the data model of OnFORME, but the functionality has not yet been implemented in the user interface.

functionality. As these features are found in just under half of the sample applications, they can be seen as product differentiators.

9.3.6 OnFORME Features Found in 2 of 7 (29%) of the Systems

There was a cohort of ten features (16% of OnFORME's feature set) found in exactly two of the seven sample applications (or 29% of the application space). These features are listed in Figure 9-7.

Survey Reusability
Student Self-Registration Option
Dropdown Question Type
Textbox Height Control
Textbox Size Control
Font Control
Question Grading (with "Right" answers)
Option to Show Respondent "Right" answers
Custom HTML Wrapper for Respondent Pages
Label Objects

Figure 9-7: OnFORME features found in 2 of 7 comparison systems

These features represent functionality that is not mainstream in the space, and they are explained below.

- Survey Reusability – A step higher from cloning (or copying) surveys, is using the same survey in more than one place. The feature allows a larger dataset to be easily evaluated for a survey, since the different survey uses are pointing back to the same survey object in the database. E.g. if an instructor teaches two sessions of the same class and wants the students to evaluate his teaching, cloning a survey for use in each class would give a result set for each class, and the aggregate would have to be manually calculated. Conversely, reusing the same survey twice would allow two distinct result sets or an aggregate set covering both sessions to be analyzed. Having more than one survey use pointing to the same survey object in the database allows for automatic result aggregation.
- Student Self-Registration Option – To have a course list of students eligible for surveys, instructors must enter those students manually. Allowing the instructor to push the registration responsibility to the students is a way to eliminate that time requirement.
- Dropdown Question Type – There are instances in which the respondent is asked to pick one of a list of options, but the number of options is so great that listing them would take an large amount of space on the page (e.g. having a user pick which of the 50 states they live in). Allowing questions to be presented in a dropdown format (in which not all options are visible at once) provides a solution to the "pick one of very many" use case.
- Textbox Height Control – The "Free Text" question type can be used in many situations, ranging from one-word answers to entire essays. By allowing

assessment creators to control the number of lines shown in a textbox, they can adapt the question to the type of information it is expecting.

- Textbox Size Control – Similar to Textbox Height Control, this feature allows the length of a text box to be controlled, to make it more relevant to the type of data expected.
- Font Control – This feature allows for control over the size of the fonts used for individual questions or blocks of text displayed on the assessment.
- Question Grading (with "Right" answers) – This type of functionality is necessary for graded assessments (e.g. quizzes), in which there must be a way to specify the "correct" answer to various questions. The potential of defining a "correct" answer option enables automatic grading, etc.[70]
- Option to Show Respondent "Right" answers – Once questions are set up to have a "Correct" answer, then a further feature becomes available for giving feedback to the student. After they answer the question, they can be told whether or not they chose the right answer (and what the correct answer is).[71]
- Custom HTML Wrapper for Respondent Pages – This feature allows those giving assessments to create their own "branding" on a survey system, consistent with the expectations of those coming to take a survey. This feature could be implemented on a server level (as is the case with OnFORME) or on a survey level (as is the case for QTools and Navigo).
- Label Objects – Besides questions, assessments often need additional pieces of text that serve to mark separations between sections, give instructions to the user, etc. The only two applications (beside OnFORME) that allowed text to be included in an assessment were QTools and Navigo.

These features have less prevalence in this application space, as only a little over one quarter (29%) of the applications considered possess them.

9.3.7 OnFORME Features Found in 1 of 7 (14%) of the Systems

There were nine of OnFORME's features (15% of its functionality) that were found in only one other application in this space. Those features are listed in Figure 9-8.

Survey Reuse Presentation Parameters
Question Reusability
Verification Login
Radio with Comment Question Type
Checkbox with Comment Question Type
Answer Alignment Option (V/H)
Required Option
Email Queuing
Rule-Triggered Emails

Figure 9-8: OnFORME features found in 1 of 7 comparison systems

[70] This functionality is part of the data model for OnFORME but has not been implemented to date due to lack of user demand.

[71] This functionality is part of the data model for OnFORME but has not been implemented to date due to lack of user demand.

Here, the functionality is far from common practice, and is almost unique, as OnFORME and only one other application share it. Descriptions of these features are included below.

- Survey Reuse Presentation Parameters – Closely coupled with the concept of survey reusability (to enable system-wide response aggregation) is the reality that the same survey may be used in different places under different circumstances. By having usage parameters (e.g. access rights, active dates, etc) distinct from the survey itself, greater flexibility of content reuse is enabled. The only other application that includes this concept is Navigo, and that functionality is only in the architectural plans and is not yet available.
- Question Reusability – This is a finer granularity then Survey Reusability with the same goal in mind. By allowing the same question object in the database to be used by more than one survey (without making a copy of the question), the data collected by that question can be aggregated across the entire feedback system. E.g. suppose an instructor has a demonstration that she uses in many different subjects and she wants to know the widespread opinion of that demonstration. This information is possible to obtain by reusing the same questions pertaining to that demonstration across the different surveys for each of the different subjects. The only application with this concept is Navigo, and although it is in their architectural specification, the functionality is not yet included in the application.
- Verification Login – Requires the survey respondents to supply a valid username and password, but that identifiable information is kept separate from the submitted response. This allows accountability for survey submission (a list of users who have completed the survey is maintained, apart from the answers) without hampering the anonymity of responses themselves. QTools was the only application (other than OnFORME) that offers this functionality.
- Radio with Comment Question Type – This is a Radio Question which also has a textbox appended to it. This enables radio questions with a "Please explain" textbox that can be used to collect additional information from the respondent. Navigo is the only other application to support this feature.
- Checkbox with Comment Question Type - This is a Checkbox Question (multi-select) which also has a textbox appended to it. This enables checkbox questions with a "Please explain" textbox that can be used to collect additional information from the respondent. Navigo is the only other application to support this feature.
- Answer Alignment Option (V/H) – Enables survey authors to control the orientation of multiple-choice options (radio or checkbox). This allows for formatting that is more appropriate for individual questions. E.g. If a question asks that a value between 1 and 10 be chosen, it would not be graphically pleasing to display 10 lines with a single character. Sloanspace is the only other application to support this feature.
- Required Option – Allows survey authors to require that individual questions be answered, and the survey cannot be submitted until all of the required questions are answered. Sloanspace is the only other application to support this feature.

- Email Queuing – Allows an email to be created and then queued so that it is sent on a predetermined date. Sloanspace is the only other application to support this feature.
- Rule-Triggered Emails – Allows emails to be sent automatically, triggered by a survey rule (e.g. the receipt of an email, or the lack of receipt of an email). Sloanspace is the only other application to support this feature.[72]

These features, representing 15% of the functional architecture of OnFORME, are rare in their appearance in this application space, owing to their appearing in only one of the seven (14%) comparative applications.

9.3.8 Features Uniquely Found in OnFORME

The most interesting part of this analysis is in identifying the features of OnFORME that do not appear in any of the other studied educational feedback applications. There were 15 features (25% of OnFORME's functionality) that are unique to OnFORME. These features are listed in Figure 9-9.

Session Based Absenteeism Control
Question Reuse Presentation Parameters
Anonymous Option
Verification Option
Attendance Based Survey Access
Radio with Textbox Option Question Type
Checkbox with Textbox Option Question Type
Dropdown with Textbox Option Question Type
Dropdown with Comment Question Type
Multi-Select Dropdown Question Type
Max Answer Limit (multi-select)
Max Answer Limit Display Option
Instructor Specific Option
Feedback Channel to Material Authors
Attendance-Based Emails

Figure 9-9: OnFORME features not found in any of the comparison systems

These features highlight novel functionality contained in OnFORME. Descriptions of these features are listed below.

- Session Based Absenteeism Control – Allows an instructor to maintain lists of those students that miss sessions. This feature enables further controls that are linked to session attendance.
- Question Reuse Presentation Parameters – When a question has been reused in more than one survey, use presentation parameters allow the question to function in different ways within different surveys. These use parameters could be visual (e.g. vertical or horizontal answer options), structural (e.g. the answer

[72] This feature is supported by the OnFORME architecture, but it has not been implemented in the interface at this time.

options formatted as Radio Buttons or Checkboxes), or functional (e.g. questions which are required in one survey, but not in another).

- Anonymous Option – Enables situations in which the survey author wishes to empower the respondent with the ability to choose whether or not they want to submit a survey anonymously.
- Verification Option - Enables situations in which the survey author wishes to empower the respondent with the ability to choose whether or not they want to submit a survey anonymously after their identify has been verified.
- Attendance Based Survey Access – There are classroom scenarios in which the survey is only appropriate for those who were in class. This feature allows the instructor to block survey access from those marked "absent" from a particular session.
- Radio with Textbox Option Question Type – Enables the association of a textbox with a radio button answer option. This allows an "Other" value to be entered, and the limit of only a single response is enforced (as opposed to Radio Button with Comment, which can accidentally return two responses).
- Checkbox with Textbox Option Question Type – Enables the association of a textbox with a checkbox answer option. This allows an "Other" value to be entered, and the multi-select limit will still be enforced (as opposed to Checkbox with Comment, which can accidentally return an extra response).
- Dropdown with Textbox Option Question Type – Enables the association of a textbox with a dropdown answer option. This allows an "Other" value to be entered, and the response limit will still be enforced (as opposed to Dropdown with Comment, which can accidentally return an extra response).
- Dropdown with Comment Question Type – Allows additional text input to be associated with a dropdown response (e.g. having a "please explain" box appended to a dropdown box).
- Multi-Select Dropdown Question Type – This allows a dropdown box object which can have more than one answer option selected. An example might be a long list of career paths, where multiple paths can be chosen.
- Max Answer Limit (multi-select) – Checkbox Questions and Multi-Select Dropdowns give the option to the respondent to select more than one answer option, and this feature allows an enforced upper limit to be set.
- Max Answer Limit Display Option – Allows the instructor to choose to display the maximum number of options that can be selected by a user.
- Instructor Specific Option – This is one of the most novel innovations of OnFORME. It allows questions to dynamically adapt to the course in which they are used. When a question is tagged as Instructor Specific, it will be repeated for every registered instructor in a course, and the response for that question is viewable only by the instructor for which it is asked.
- Feedback Channel to Material Authors – This feature is intended to allow feedback to flow to third parties (other than instructors) involved in the educational content creation process.[73]

[73] The foundations of this functionality are embedded in the OnFORME architecture, although much work is required to make this easily usable.

- Attendance-Based Emails – This feature allows emails to be sent to a class while excluding those who did not attend. It allows pertinent information to be sent to those to whom it matters.

It is important to note that while these features are not all major advances (e.g. showing the maximum number of allowed selections), they do represent functionality that is not apparent in the other applications considered for comparison. This could be due to a lack of demand from the customers of these other systems. Or it could be indicative of an underlying object model that was set in place without this functionality, and is now difficult to change. Either way, these features represent new insights of varying value concerning application functionality in this space.

9.4 Conclusions

Based on a rough comparison of the sample of applications found in this space, it is safe to conclude that broadly speaking, the OnFORME architecture does not merely mirror the common functionality found in the educational feedback applications. This is evidenced by:

- 24% of the enumerated features in the OnFORME architecture are not found in any of the other comparison applications.
- 39% of the enumerated features in the OnFORME architecture are found in no more than 1 of the 7 comparison applications.
- 55% of the enumerated features in the OnFORME architecture are found in a little over a quarter (29%) of the comparison applications.

This information is shown in Figure 9-10.

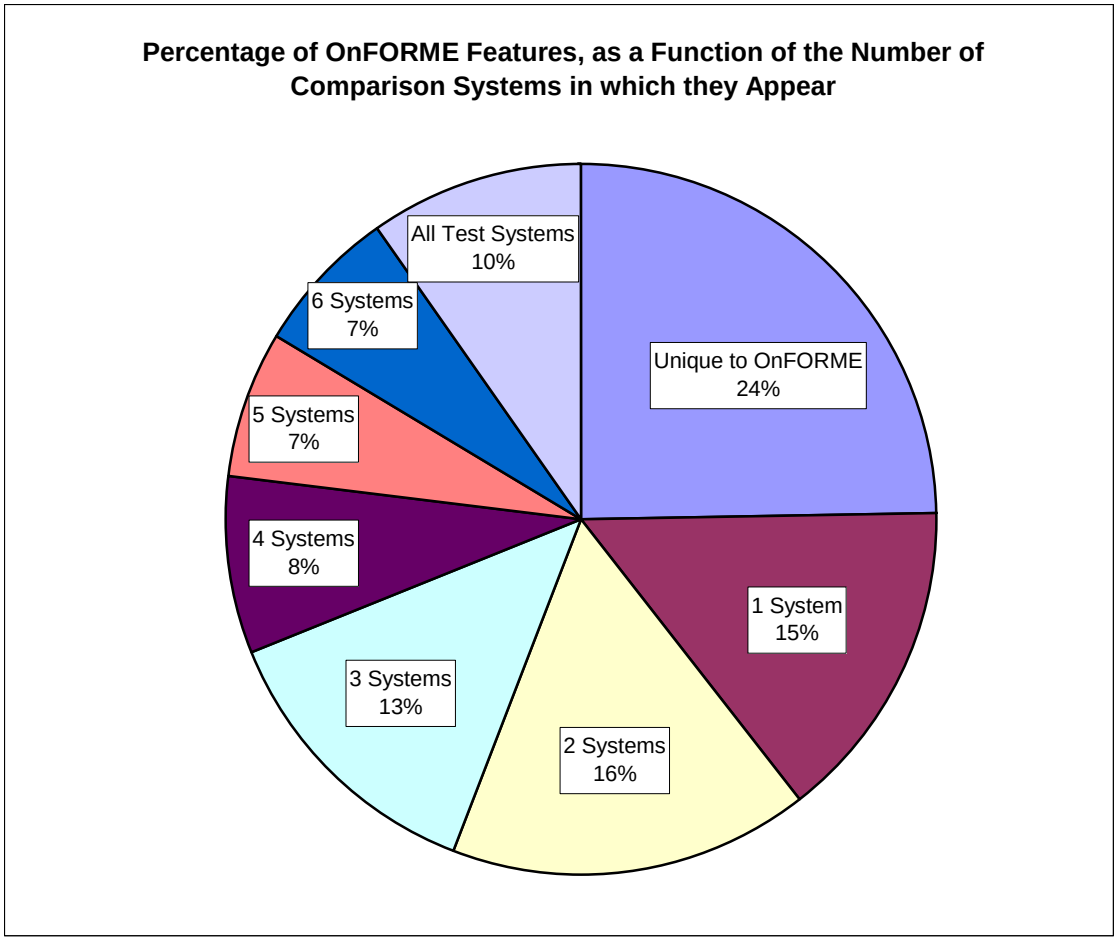


Figure 9-10: Summary of Results from Competitive Analysis

The novelty of many OnFORME features was further reinforced through communications with users and designers in this space. For example, several times during the interview with Lance Speelmon (the leader of the 16-person joint-university Navigo design team), he responded with interest when presented with questions concerning certain features contained in OnFORME. When the rationale (use cases) behind these features were explained, there were several times where he answered by saying that this was a use case which they had not considered, and he would go back and discuss with his designers the feasibility of adding that feature. In another instance, the object model underlying OnFORME was described to the head developer from a general use web-polling company (PulsePoll.com), who responded with enthusiasm and began to inquire about who owned the rights to the work, and whether or not the model might be available on a licensed basis.

While web surveys have been around for over a decade, it is clear that applications enabling non-technical instructors to easily create and publish their own assessments is an area where there is current growth and future potential. This is especially evident by the joint university team from Stanford and Indiana University that have recently entered this space. This is a place where advancements can be made, and the architecture of

OnFORME certainly appears to contain innovations that add value in this application space.

Part III – Social Systems Components

10 Description and Analysis of Usage Trials

The OnFORME system has undergone a number of usage trials to date, covering a variety of locations and types of use. This chapter groups those trials into like categories, describes the type and style of use, and then discusses the results of the usage trial. This is intended to be descriptive in nature, leaving the analysis of these usage trials to the two chapters following this one.

10.1 Beta Trials

Two Beta Trials were performed at MIT during the fall semester of 2003. These trials took advantage of the willingness of early adopters to try out the system despite its shortcomings as a beta release. The descriptions of both of these trials and their results are shown below.

10.1.1 ESD.801 Leadership Weekend

10.1.1.1 Description

The entering Technology and Policy Program (TPP) graduate student cohort at MIT is required to take ESD.801, which is a leadership development course. Part of this course is an offsite Leadership Weekend comprised of team building events, leadership training, and a ropes course. The TPP administration wished to gauge the effectiveness of the weekend in accomplishing its goals, and to solicit student opinion as to the merits of various aspects of the event. They put together a survey inquiring about these topics for the ESD.801 Leadership Weekend. This was a one-time survey of this group concerning this topic.

10.1.1.2 Results

As an incentive to get students to respond to the survey, the students were told that when they finished the survey they would be brought to a page containing the digital pictures from the leadership weekend. An email with the survey URL was sent to the students a couple of weeks following the leadership weekend.

The table in Figure 10-1 contains the results from the ESD.801 Leadership Weekend Beta trial. The response rate to this first beta trial of OnFORME was 64%.

Subject	Session	Submission Mode	Login Mode	Associate Response	Total Respondents	Size of Respondent Population	Response Rate
ESD.801	Leadership Weekend	Single[74]	Login Required[75]	Yes[76]	29	45	64%

Figure 10-1: Highlights of the ESD.801 Leadership Weekend Beta Test

Aside from the raw numbers, this trial had the result of helping to identify several usability issues with OnFORME, both from a respondent and an administrative standpoint (discussed in the next chapter).

10.1.2 ESD.801 Leadership Seminar Review

10.1.2.1 Description

ESD.801 holds a concluding seminar session toward the end of the semester. The course instructor (Dr. Cutcher-Gershenfeld) wished to assess the learning of the students and the value of the last session and its contents. He therefore set up a survey to ask the opinion of the students on a variety of topics concerning the final ESD.801 session. This was intended to be a one-time surveying of this course on this topic.

10.1.2.2 Results

The second ESD.801 beta test did not include any sort of incentive structure. The announcement went out to the course students via email during the last couple of weeks of the semester. The result was a low response rate of 31%

Subject	Session	Submission Mode	Login Mode	Associate Response	Total Respondents	Size of Respondent Population	Response Rate
ESD.801	Concluding Feedback	Single	Login Required	Yes	14	45	31%

Figure 10-2: Highlights of the ESD.801 Seminar Beta Test

An interesting point of reference surfaced a couple of months later when I interviewed some students concerning this second survey. Many of those with whom I spoke did not answer the survey, and in fact they did not even recall receiving an email about this second survey (they remembered the first one). This raises the possibility that there were

[74] Single Submission mode means that each student is only allowed to submit one response to the survey.
[75] When the Login Mode is set to "Login Required", it means that each student must submit a valid username and password in order to access the survey.
[76] Setting Associate Response to "Yes" means that each response will be tagged with the name of the respondent who submitted it.

disconnects in communication (possibly the timing or form), which had an impact on the response rate.

10.2 MIT Production Trials

Following the two beta trials at MIT, two courses used OnFORME on an ongoing basis for the spring semester of 2004. These were both courses within the TPP curriculum, and their ongoing nature made for a different type of usage trail from the other test sites that were one-time events.

10.2.1 ESD.140 Organizational Processes

10.2.1.1 Description

ESD.140 is a course required for each graduate student in the TPP program. It is taught by Dr. Joel Cutcher-Gershenfeld, who is very interested in student learning and pedagogical improvements. Dr. Cutcher-Gershenfeld decided that he wished to use OnFORME to collect weekly feedback on the students' learning and their opinions of the lectures. He sent out a series of 11 weekly surveys asking three basic text entry questions on the weekly lecture.[77] Toward the end of the semester he sent out one longer survey requesting feedback on a Sloan System Study[78] used during a single lecture.

10.2.1.2 Results

In order to provide incentive for the ESD.140 students to take the time to submit weekly feedback, Dr. Cutcher-Gershenfeld is basing part of each student's class participation grade on how many of the surveys they respond to. The incentive had its desired result, as can be seen in Figure 10-3.

[77] There were more surveys that went out after the data cutoff date for this thesis.
[78] Sloan System Studies are learning objects created for the purpose of teaching key concepts within the emerging field of Engineering System.

Session	Total Respondents	Size of Respondent Population[79]	Response Rate
1	35	35	100%
2 & 3	33	35	94%
4 & 5	30	34	88%
6 & 7	31	34	91%
8 & 9	27	34	79%
10 & 11	31	34	91%
12 & 13	28	34	82%
14 & 15	30	34	88%
16 & 17	28	34	85%
18 &19	31	34	91%
20	28	34	82%
System Study	16	34	47%

Figure 10-3: ESD.140 Weekly Survey Response Data

Login Required; Responses Associated with Respondents

The response rates for the ESD.140 surveys were all quite high with the exception of the System Study survey. The average response rate for the session surveys was 89%.

10.2.2 ESD.80 TPP Thesis Seminar

10.2.2.1 Description

The ESD.80 TPP Thesis Seminar is a course in which TPP students give a presentation on their research prior to graduation. Part of the course is a peer assessment evaluation, in which one or two students in the class review each presentation. The course instructor, Prof. Joel Clark, agreed to use the OnFORME system as a means of collecting the evaluations electronically. In previous years, the evaluations were completed via paper submission. The evaluation consisted of several multiple-choice questions assessing the

[79] One of the students dropped the class after the second survey.

quality of various aspects of the presentation, with the option of giving additional comments to the presenter.

10.2.2.2 Results

This trial ended up being more difficult to conduct than what was initially assumed. Due to the direction of flow of the feedback (from student to student), the educational structures designed within OnFORME were not able to accommodate the automation of this process. In order to automate this in the paradigm of OnFORME, it would have been necessary to create a separate course for each presentation, assign the presenter as the instructor, and then make all of the questions instructor specific so that other students entering the system could not read the reviews made of their peers. There were two problems with this approach:

- The OnFORME system does not have the capability of batch-creating courses, and creating 36 distinct courses, sessions, and survey uses would have been very time intensive.
- The window of opportunity between this trial being confirmed and the date of the first presentation was too small to allow the "Instructor Specific" functionality to be programmed into the system. By the time this functionality was fully implemented, the ESD.80 trial had already been proceeding for a month.

Given these constraints, another approach was used for this trail. This approach was simpler, but required a much greater level of manual interaction to make it work. There was a single course and a single survey used, and the survey included a free text question asking for the name of the presenter. The recorded responses were manually separated, and then sent to the presenters via email. These steps added a considerable amount of administrative overhead to the use of OnFORME for ESD.80. The amount of extra time required made it questionable whether or not the use of the system was better from a course management perspective than the previous paper-based system. Less administrative time was spent cumulatively using OnFORME, but the heads of the course spent less time with the old system. With the paper-based system, the administrative efforts were delegated to the evaluators who had the task of duplicating their work and giving it to both the presenter and the course instructor. So, although OnFORME allowed statistical analysis across the entire course, the additional efforts to distribute the evaluations proved to be inhibitive.

The basic use statistics are shown in Figure 10-4.

Subject	Submission Mode	Login Mode	Associate Response	Total Responses	Size of Respondent Population
ESD.80	Multiple[80]	Login Required	Yes	69	32

Figure 10-4: Statistics for ESD.80 Usage Trial

Every student submitted at least one survey, giving an overall response rate (or involvement rate) of 100%. On a presentation-by-presentation basis, this would be very hard to reconstruct the specific response rates (i.e. figure out of the 2 or 3 students assigned to that presentation did in fact evaluate it). This would be difficult because the evaluators sometimes had to swap assignments due to schedule conflicts. Additionally, with such small sampling sizes the results would not be helpful. Therefore the total number of responses, and the learning surrounding the adoption and use of the system in ESD.80 (as discussed in the following chapters) were the key data outputs of this trial.

10.3 Cambridge University Trials

A portion of the funding for research using OnFORME came through the Cambridge-MIT Institute (CMI) and thus there was an interest to explore ways in which this application could benefit the University of Cambridge. Specifically, CMI wished to see if the new interdisciplinary M.Phil programs at Cambridge could use OnFORME. I was sent to Cambridge University to interview potential users, explain the functionality of OnFORME, and offer whatever services I could to try to support the teaching and administrative efforts within the M.Phils. The end results of that visit were:

- Two immediate usage trials (see the following sub-sections)
- Two planned usage trials
  - TP4 Classroom Educational Exercise review (to occur in May 2004)
  - TP4 term-end review (to occur in May 2004)
- One potential usage trial
  - Instructors in the Sustainable Development M.Phil were interested in exploring how OnFORME might benefit their program but as of the writing of this paper have not followed through on that interest.
- Adoption of OnFORME on a trial basis for all of the CMI M.Phils
  - CARET – the technology support service at Cambridge University – has decided to install OnFORME and use it on a test basis as a possible permanent feedback support solution for the CMI M.Phil programs.

The interest shown in the OnFORME system was quite encouraging and the prospect of CARET joining the development effort is a welcome one. Given that overview, it is

[80] As the same survey was used for the entire semester, and each student submitted multiple reviews, the Submission Mode of this survey was set to "Multiple".

instructive to now discuss the actual Cambridge trials of the system to date, and their results.

10.3.1 5CMI3 Logistics Systems Control

10.3.1.1 Description

5CMI3 Logistics Systems Control was a Lent term course taught by Prof. Bill Nuttall and Prof. John Ash. This is a fairly new course, and student feedback is helpful to improve and shape it in a way that best meets the course objectives and addresses students' needs. Prof. Nuttall wished to use OnFORME to draw out details of the usefulness of the course, and find ways in which the students felt that it could be improved.

10.3.1.2 Results

The 5CMI3 cohort was a small one, and about half of the students were auditing the course. An email went to the students several weeks after their last lecture, asking that they take a few minutes and comment on the course. There was no incentive associated with filling out the survey. All responses were collected anonymously. The results of the 5CMI3 usage trail are shown in Figure 10-5.

Subject	Session	Login Mode	Total Respondents	Size of Respondent Population	Response Rate
5CMI3	Post-Module Evaluation	AlwaysAnonymous	5	8	63%

Figure 10-5: 5CMI3 Usage Trial Results

Considering the lateness of the request for feedback and the number of auditors, 63% was a very good response rate.

10.3.2 Management of Technology and Innovation

10.3.2.1 Description

The Management of Technology and Innovation (or MOTI) is a brand new module taken by students in three different M.Phils. The module includes a series of lectures, as well as an extensive project performed with a corporate customer. It included 69 students, broken down by program as follows:

- 19 from the Bioscience Enterprise M.Phil
- 23 from the Engineering for Sustainable Development M.Phil
- 27 from the Technology Policy M.Phil

As this was the first time that the module was offered, Prof. Nick Oliver was very keen on trying to capture student option, especially concerning the corporate project. Some of the students from the class were also pushing for a survey, as they felt that there were things that should be changed to benefit the following year's students.

10.3.2.2 Results

Prof. Oliver opted to make this an anonymous survey to save the respondents the hassle of having to register and identify themselves. An email with the survey URL was sent to the students a few days after they turned in their final paper. There was no explicit incentive offered - it was merely posed as a sincere request for feedback from the students. The results of this test are shown in Figure 10-6.

Subject	Session	Login Mode	Total Respondents	Size of Respondent Population	Response Rate
MOTI	Post-Module Evaluation	AlwaysAnonymous	16	69	23%

Figure 10-6: Results of MOTI Usage Trial

The MOTI trial collected a wide variety of opinions from the students concerning the module and how it could be improved.

10.4 InterSystems Corporation, Learning Services Trials

The corporate trials of the OnFORME system were performed at InterSystems Corp. within the structure of their classroom-training curriculum. There are two basic structures of training which InterSystems provides:

- On-Site – For these courses, an InterSystems trainer travels to a customer location and gives the course "On-Site" (the customer's site).
- Local – These courses are taught at the InterSystems corporate headquarters, and customers travel to Cambridge, MA to receive training.

At the InterSystems trial, the same survey is given for all courses, and the survey contains a number of "Instructor Specific" questions to tailor each survey to the trainer who taught that course.[81] Every survey use required the students to register themselves, and then gave them the option of whether or not to perform a "Verification Login"[82] to take the survey. This section compiles the results of the InterSystems trial, grouping the evaluations by the manner in which they were given.

10.4.1 In-Class Evaluations

10.4.1.1 Description

Following the conclusion on a course (either On-Site or Local), it is preferable for the trainers to give an evaluation before the students leave the training room. This provides a

[81] Instructor Specific questions, as described in previous chapters, adapt themselves to the instructors registered to teach the course to which a survey is attached. They therefore are dynamic questions that become context-sensitive.

[82] A Verification Login, as described in previous chapters, is when a use must provide valid login information, and then their response is recorded as "Anonymous" within OnFORME.

captive audience, and it is believed that there is a higher likelihood that students will follow-through and take the time to complete the survey (this is a question examined more closely in the following chapters).

10.4.1.2 Results

At the time of the writing of this thesis, three courses had been completed in which the evaluation was given during the last course session. All of these were On-Site courses. The results of these trials are shown in Figure 10-7.

Course	Instructors	Length of Course (days)	Total Reponses	Associated Respondents[83]	Anonymous Respondents[84]	Size of Respondent population	Response Rate
Systems Management - Sue, VA NW	O'Leary, Sue	3	12	10	2	12	100%
Cache System Management	O'Leary, Sue	3	4	2	2	4	100%
Cache Core: Server-side Programming	McDaniel, Adele	5	12	9	2	15	80%

Figure 10-7: Results of In-Class Evaluations

Login Required; Students' Choice for Verification Login

There was an extremely high response rate for the evaluations that were given before the students left the training room.

Concerning the course taught by Adele, there was a considerable amount of user error caused by the instructor not understanding the student login procedures. This caused a large amount of confusion among the students, and an 80% response rate was actually quite good. The lessons learned from this botched-trial are discussed in the following chapters.

10.4.2 Post-Course Evaluations

10.4.2.1 Description

During the trial period, some of the courses did not distribute the evaluation during the last session. Instead, the survey was sent to the students via email some number of days after the course had concluded. The evaluations were given late for a number of reasons:

[83] Associated Respondents are those that choose to associate their login information with their submitted response.

[84] Anonymous Respondents are those that choose to not associate their login information with their submitted response.

- Timing - The OnFORME system was not ready for use at the time of the last session.
- User Error - The instructor lacked the sufficient knowledge to successfully direct the students to the appropriate OnFORME survey.
- No Access - There was no Internet access available in the training room, so the students could not access OnFORME from class.
- Habit - Out of habit, the instructor did not think to do the survey during the last session.

The late evaluations had the same access parameters as those given in-class.

10.4.2.2 Results

There were six courses (all On-Site) for which the evaluations were distributed late. The results of these feedback requests are shown in Figure 10-8.

Course	Instructors	Length of Course (days)	Reason for Evaluation Delay	Days until Survey Emailed[85]	Total Reponses	Associated Respondents	Anonymous Respondents	Size of Respondent population	Response Rate
Caché Core: Server-side Programming - DHMC	Solon, Joel	4	Timing	3	5	5	0	11	45%
Introduction to Caché-Niederhoffer	Solon, Joel	1	Timing	11	0	0	0	7	0%
Caché Core Server-Side Development - Adele – Morning	McDaniel, Adele	10	User Error	7	3	2	1	10	30%
Caché Core Server-Side Development Adele – afternoon	McDaniel, Adele	10	User Error	6	3	2	1	10	30%
DevCon: Introduction to Caché Sytem Management	O'Leary, Sue	2	No Access	3	5	5	0	8	63%
Caché Server Pages – DHMC	Solon, Joel	3	Habit	3	1	1	0	10	10%

Figure 10-8: Results of Post-Course Evaluations

Login Required; Students' Choice for Verification Login

The results from the follow-up evaluations show clearly that students are less responsive when given a survey after class, than they are when the survey is part of their class-time. This theme is explored further in the following chapters.

[85] This represents the period of time between the last course session, and the day when the email went to the students asking them to fill out the survey.

Both of Adele's courses had the evaluations sent after the course due to errors caused by her unfamiliarity with OnFORME. In each case, she did not read the instructions detailing how students are to access the system, which resulted in confusion in trying to administer the surveys at the end of the last session. Both times, she defaulted to using an older survey system at the end of class, and then she sent an email a week later asking that students enter their responses again into OnFORME. Given these details, a low response rate for both of her courses was not unexpected.

11 Domains of Learning – Adoption

The time and effort spent attempting to simulate adoption and use of OnFORME in various settings provided valuable insights into organizational processes and user behavior. As mentioned in the beginning of this thesis, the data collected was intended to generate inductive hypothesis that can be later tested. This was approached through case study analysis, as opposed to structured experimentation.[86]

This chapter presents the learning that took place through attempting to enable the adoption of OnFORME for various usage trials. The learning could be separated into two groups, which represent the two hypotheses that I would like to put forth as a result of the findings:

- ❖ User pull is strongest when the new system is an improved replacement for an existing process, and that process is well understood.
- ❖ When instructing new users in the use of a new IT system, being there in person and teaching by demonstration increases the likelihood of adoption and use.

The following sections flesh out these hypotheses and present the mini-lessons learned through the adoption process of OnFORME.

11.1 User Pull and Replacement of Existing Processes

From its onset, the OnFORME development effort had guaranteed trial sites through its customers. The challenge was finding additional sites in which the system could be tested in order to collect more use cases, and more usage data. There were a number of possible trials in which the usage was brainstormed and the details were discussed with the potential users, but only two-thirds resulted in an actual trial case. Of those trials that did occur, many were more the result of developer push than customer pull, and therefore will likely not lead to continued use of the system once the current trial is complete (and the developer graduates). A breakdown of the usage trials categorized by their perceived driving force is shown in Figure 11-1. This table also lists trial codes, which will be used as references to the specific trials throughout the chapter.

[86] This route was chosen due to the limited amount of time remaining following the construction of the OnFORME system. Had the system been available from the onset, it could have been used to collect more extensive use data for deductive research. Hopefully, future students can proceed from this starting point.

Trials Primarily from Developer Push	Trials Primarily from Customer Pull
DP1 - ESD.801 Leadership Weekend	CP1 - InterSystems Classroom Training
DP2 - ESD.801 Final Session[87]	CP2 - ESD.140 Weekly Surveys
DP3 - ESD.80 Thesis Seminar[88]	CP3 - TP Module Evaluations[89]
DP4 - 5CMI3 Evaluation	CP4 - MOTI Evaluation[90]
	CP5 - CARET Adoption[91]

Figure 11-1: Breakdown of the Driving Forces Behind the OnFORME Trials

A common factor in these usage trials is that each user had some level of perceived value from using the system that was greater than the usage cost. The CP (Customer Pull) trials were the ones in which the value was perceived to be the greatest, which resulted in the customer taking the initiative to pursue using the system. For the DP (Developer Push) trials, the customer had to either be convinced of the value, or the value was altruistic (in being a willing trial site for graduate thesis research).

It is also instructive to consider the potential trials that never materialized. These trials are listed in Figure 11-2 (these trials will be individually referred to by their FTx label in further discussions).

Failed Trial Attempts
FT1 - InterSystems eLearning
FT2 - Introduction to Aerospace Engineering
FT3 - TPP Curriculum Review[92]
FT4 - CMI Sustainable Development MPhil

Figure 11-2: OnFORME Trials that Did Not Occur

These trials failed for a number of reasons, but most of them due to the amount of push that would have been required to get them off of the ground. It is likely that given more of a time investment by myself, it would have been possible to launch most of these

[87] The customer initiated this trial, but there was no follow-through to retrieve the feedback collected by the trial. This points to a possible motivation other than that of wanting the data (in this case, the motivation is assumed to be that of wanting to be of service as a trial site for this research).

[88] This trial required only a minimum amount of push to be approved, as the instructors did not care if OnFORME was used to automate their system, provided it didn't mean any more work for them. This translated into their not engaging the tool at all (except for an initial overview with one of the two instructors). Therefore the push in this trial was primarily in doing all of the work to run the trial.

[89] This trial will be taking place after May 17th.

[90] This Pull was actually an interested third party (students who had just finished the class) as opposed to the actual customer of the survey information (the module instructor).

[91] This has not occurred yet, as CARET's intent is to do an installation of the system at the University of Cambridge, which is a slower process than using an existing installation for a usage trial.

[92] At the time of this writing, a development took place that will likely move the TPP Curriculum Review into the Customer Pull category. One of the people responsible for the review was starting to do an informal paper survey when I reminder her about OnFORME (she had been interested several months earlier). She quickly reinstated her interest, and plans on using it to save time over the paper-based system. At the time of this writing no action had yet occurred, but it looks likely that she will follow through and initiate use of the system.

trials. However, given the existing amount of use generating data for this thesis, capturing trials requiring more push than those already underway was not a high priority.

Prior to discussing in detail an analysis of lessons learned surrounding push, pull, process and adoption, a mini-case that highlights many of these principles is presented. Throughout the mini-case and in the following sections, points of insight that I gained are inserted into the narrative, numbered and marked with a light bulb (). The trials that provided and reinforced these insights are listed parenthetically following the explanation of the learning.

11.1.1 Cambridge Technology Policy Mini-Case Study

I took a research trip to Cambridge University in March of 2004. This trip was for the express purpose of talking to the instructors of the new CMI MPhil programs and finding ways in which OnFORME could be leveraged within the new programs. I quickly discovered that professors at the University of Cambridge are almost identical to their peers in the U.S.

1. University Professors are very difficult to capture as adaptors because they are too busy to learn new processes or try new applications. (DP4, CP4, FT2, FT3)

I was there for several days, and it was very difficult for me to successfully connect with professors in a substantial way that resulted in user trials. The exception to this was Prof. Bill Nuttall, the Director of the Technology Policy M.Phil. Given a previous working relationship with Prof. Nuttall, he was my main point of contact. He was very generous with his time and advice on how I should approach finding users at Cambridge. He himself was very supportive of my research, but he was honest about the restraints placed on his time, and the small likelihood that he would be using OnFORME extensively.

Prof. Nuttall did me a great service by taking the time to explain the ways in which OnFORME might be used at university of Cambridge. There were really two very different reasons for feedback collection that existed, even within his department.

2. The need for feedback can differ within the same organization between different roles. (CP3)

My focus had been entirely on the course instructors, and their use of the system for the improvement of course materials and teaching methodologies. He explained that, in accordance with Insight 1, the instructors would be difficult to persuade because they were so busy (which I certainly found to be true). He also pointed out that there was another motivation for feedback within the department which was not instructor driven. This was the administrative motivation to collect regular feedback for the purposes of monitoring the performance of its instructors.

3. Feedback can be instructor driven for self-improvement or administration driven to track instructor performance. (CP1, CP3)

He explained that at the University of Cambridge, there is a sharp divide between the academic staff and the administrative staff, and that processes often run parallel without intersection. The formal evaluations given at the end of a course are run entirely by the administrative staff and the instructors are not stakeholders in this system. This evaluation process is a time-consuming part of the administrative staffs' job, and they are required to do it by the administration of the school of Management (in which the TP MPhil is housed).

4. Understanding the motivation behind why a users needs feedback is important to properly empower the adoption. (CP3, DP3)
5. Understanding the vested interests in existing feedback processes helps to identify potential users. (CP1, CP3, DP3)
6. Knowing what party is driving the feedback process in an organization will help focus on the proper targets for adoption. (CP3)

Professor Nuttall's advice led me into conversations with Paula Sparling, the Course Administrator for the TP MPhil. Paula was extremely busy, but was willing to discuss the evaluation process with me. She explained that while their current process wasn't perfect, it was at least functioning and the last thing she wanted was to try to squeeze the implementation of a new system into her already tight schedule. In fact, this was the exact same response that I received when I tried to introduce OnFORME to the administrative staff member at InterSystems.

7. Often, people don't see the shortcomings of their existing system and would rather stay where they are than make time to learn a new process. (CP1, CP3, FT1)

After some explanation and a demonstration, I was able to show Paula that using OnFORME would not result in another thing that she had to "squeeze in," because it's use would free up more time than it required (see the motivating example in the first chapter). After I explained the way in which OnFORME could replace her current process while giving her the same result she needed to deliver to her superiors, she became very enthusiastic and wanted to know when she could start using it. This was a perfect example of the power of "user pull." Once the merits of OnFORME were properly communicated in a way that targeted the user's current needs and processes, Paula was ready to take the process and move it forward without my prodding.

Besides exploring the opportunities within the TP MPhil, Prof. Nuttall also strongly suggested that I show OnFORME to the researchers at CARET. There, I was surprised by the excitement and warm reception that my tool generated. Not only did they want to try it out, but they also wanted to know if it would be possible for them to join the research project, get a copy of the code and install a version for testing purposes at Cambridge. They also wanted to know how quickly it was possible to start moving towards this goal. It wasn't until after we were well into the discussion that I learned that

they were currently in the middle of a "build vs. buy" analysis for feedback tools, as many instructors had been requesting feedback services for their courses. My timing couldn't have been more perfect.

8. Enabling adoption is often as much opportunistic as it is strategic (CP5)

This partnership was something that I practically fell into, and it is possibly the most valuable connection for the system going forward. This highlights the fact that finding the ideal adopter is sometimes a stochastic process. It also showed the value in acting on networking advice (i.e. Prof. Nuttall's direction to contact CARET).

There was another discovery of interest that surfaced in many of the Cambridge interactions as well as multiple additional trials. Instructors would often say "yes" to the concept of using the system, but then would not know where to begin, and subsequently would do nothing.

9. Instructors cannot use a feedback collection tool unless they first know what they wish to ask and how they wish to ask it. (DP4, FT2, FT3, FT4, CP4)

More than anything, this was a reflection on Insight 1, where the instructors would not put in the time sufficient to make their desire to use the system actionable. In the case of DP4 and CP4, the trials only went forward because I was able to push the instructors just enough to decipher what it was that they wanted to collect, and then draft a survey for them to use. This provided enough momentum that they were able to engage the process and polish the survey to their liking, and the trial could move forward. The danger in those situations is that unless they find the experience to be extremely valuable and rewarding, they are unlikely to engage the process next time when they do not have someone creating the initial momentum towards use.

Finally, the last area of learning from my trip involved my interaction with staff from the Sustainable Development M.Phil. I met with two of them for a demo of the system. During the demo they expressed excitement in the functionality and seemed very interested in exploring ways in which it could improve their courses and teaching. Following my departure from Cambridge, that excitement translated into a complete lack of engagement despite my effort to help move the process forward. This also was a theme with FT2 and FT3.

10. The desire for a change is not sufficient; there must be a pressing need that drives adoption. (FT2, FT3, FT4)
11. Just because people are excited by a demo, it doesn't mean that they will take the time to follow-through and adopt the system. (FT2, FT3, FT4)

I was amazed how such a demonstration of excitement and good will could result in a total lack of action. Again, this pointed to the difficulty of engaging very busy people to the point where they are able to introduce a new process into their already packed

schedules. When meeting face to face, they are free to dream of its usefulness. However, when they return to implement it in light of the reality of their current time pressures, the new improved process becomes a casualty of the "tyranny of the urgent" and falls by the wayside.

There is a slightly different result when a potential user can be convinced that the new system will improve the efficiency of an existing process and therefore save them time. In that case, the potential user will do all that they can to put the new process in place. This was highlighted very well in the difference between trying to set up a trial in the academic side of the TP program (where there is no imperative for feedback) versus the administrative side (where feedback is a required process).

11.1.2 Lessons Learned concerning Process and Pull

The previous discussion was of the efforts to encourage adoption at Cambridge University. That mini-case provided a context within which many of the process-related insights concerning adoption could be discussed. The following section lists the additional insights that were identified concerning the interaction between process, adoption, and motivation. These insights are grouped logically by their focus, and then followed by brief comments that highlight the circumstances surrounding the learning.

12. The system must be able to fit into the existing processes of an organization with minimal change. (CP1)
    a. When multiple stakeholders are involved in a feedback collection process, it is important that the process imposed by the new system match the old process as much as possible to avoid substantial confusion of those involved. This was certainly observed at InterSystems, where the feedback process directly involved four different people.

13. Early adopters, focused on the same goals as the primary use cases, are the most valuable to the ongoing development effort. (DP3, CP4)
14. Some early adopters can have a negative effect on development if a large amount of ongoing handholding or specialization is required. (DP3)
    a. Finding users who most closely match the primary use cases allows the intended function of the system to be the focus of testing. In addition to the initial customers of OnFORME, the MOTI trial served this purpose quite well.
    b. When the use cases are not kept in mind while choosing usage trials, the trial can actually detract from the development effort. As discussed earlier, the ESD.80 trial ended up being only marginally helpful in some ways to this research, and a hindrance in other ways. The direction of the feedback did not line up with the primary use cases, and therefore a large amount of manual effort was required on my part to make the trial run smoothly.

15. Unless the responses are extremely important to them, people can forget that they asked for feedback. (DP1, DP2)
    a. When a trial is the result of development push, the user may not be invested enough in the process to follow through with collecting the results. This was the case with one of the beta trials, which took place primarily to use test the system in preparation for a production release. The "customers" for one of the first two trials did not follow up and retrieved the results that were collected for them by the system. This highlights how much these trials were the result of "push" rather than "pull."[93]

16. Beside the data to be stored and the customer use cases to be met, the processes surrounding implementation are critical to understand in order to property architect a system. (CP1, CP3, FT1)
17. Paying attention to the detail in process is vital to providing usability. (CP1)
18. Some processes are not readily apparent, and do not immediately follow from the use cases. (CP1)
19. Unless the processes are encapsulated in the design, use of the system could range from clumsy to unworkable. (CP1, FT1)
    a. Process is not always well understood by the client of a system, and therefore it may not be communicated during the use case analysis process. Two clear examples of this became evident during the implementation of OnFORME with the InterSystems classroom training.
        i. During my use case interviews with the administrative assistant who would be responsible for entering and maintaining the data in OnFORME, I was asking the question "what do you need it to store?" During the implementation of the system, it became evident that I should have also asked, "By what process do you normally enter data?" The navigation between data entry interfaces was not set up in a way that enabled her process, so it frustrated her when she tried to adapt to the system (Insight 19). I was able to correct this by studying her process, and adding links between the appropriate pages to match her workflow.
        ii. During all of my user interviews, the process by which a student accesses OnFORME for a post-class evaluation was never explicitly discussed. The beta, MIT and Cambridge tests did not raise this as a concern because each of those processes included e-mailing students a link that they follow to the system. During classroom training at InterSystems, the instructor directed the students to the evaluation by putting the URL on a slide and asking

[93] It is also possible that this is the result of miscommunication. If the customers interpreted the beta tests to be fielding a "service" (which does the work for them) rather than an "application" (with which they must interact), then it is possible that they are waiting for me to deliver results to them (something which I intend to do either way).

them to type it in. Owing to the fact that the URL used to access surveys is very long and case sensitive, the first several attempts at using it at the end of a class resulted in various levels of confusion and failure. This problem was easy to create a workaround for, but it could have been avoided entirely. The problem would not have surfaced if the student access process for classroom training had been thoroughly thought through prior to the first "live" use (Insight 18).

20. Adoption cannot occur unless the structures to support sub-processes are either included in the system, or can be integrated with the system. (FT1)
21. Some parts of an existing process must stay in place, and adding new processes to them can block adoption. (FT1)
    a. From the beginning, the intention had been to incorporate OnFORME surveys into the eLearning program at InterSystems at the same time as the classroom training. However, while attempting to get that process rolling, the efforts came to a grinding halt. It turns out that there is a massive amount of bookkeeping that supports the eLearning efforts at InterSystems. OnFORME was prepared to be the tool to maintain those books (courses, instructors, students, etc), however the parallel process of supporting the eLearning software had not been considered. The Centra server that handles the eLearning broadcasts must also have a full set of course, instructor and student records in order for it to run the sessions properly. Maintaining two sets of course records was not an option. Therefore the eLearning adoption of OnFORME has been delayed until a way to integrate the two systems can be developed (which is planned for this summer).

11.1.3 Pull and Process Conclusion

All of the insights presented in this section support the following hypothesis:

- ❖ User pull is strongest when the new system is an improved replacement for an existing process, and that process is well understood.

11.2 Leading By Demonstration

In additional to the learning concerning strategic positioning for adoption, many of the use trials highlighted the fact that there is a right way and a wrong way to train early adopters. Proper training can help to create a more positive impression of the use of the system, which can impact the likelihood of a second trial. Alternatively, if the learning process is a frustrating one, it might negatively impact their desire for use past the present trial, or even cut short their willingness to follow through with the usage trial.

11.2.1 Initial Failures at Training

In many one-on-one training situations, it is possible to help the learner by "leading" them to the right answer. Asking leading questions or inserting sufficient vagueness in the instruction causes the student to be stretched in his or her understanding and can reinforce the learning. This is my preferred method of individual teaching. However, through the experience of instructing new adopters in the use of OnFORME, I learned that this style of teaching can be more frustrating than helpful. In fact, one of the key things that I learned early in the adoption process, was that an opposite approach to training is necessary with the adoption of new technology.

22. New paradigms must be spelled out explicitly, rather than trying to "lead the witness".[94] (CP1, CP2, DP1, DP2, DP3)
23. Without a comprehensive understanding of the architecture, nothing is intuitive to the user unless a detailed explanation is given. (CP1, CP2, DP1)

Without a frame of reference to draw upon, users would not gain any traction trying to perform functions within OnFORME. After several failed attempts, I realized that my method was doing more to frustrate my users than endear the system to them. After having this insight, I changed my approach for future training sessions.

11.2.2 Responsiveness in Later Trials and Further Learning

Upon realizing that users need a mental framework in order to work their way through a new system, I began training by demonstration. This required more time, but I found that investing one-on-one time was often the most effective way to persuade a potential user to engage the system.

24. Face time is sometimes required to get user buy-in for a new system. (CP3, CP4, FT4)

This was particularly true with the Cambridge University trials. Prior to my visit, there had been some interest expressed by Bill Nuttall for a potential trial use of the system. However, had I not visited the university in person, then I most certainly would not have captured my most useful Cambridge trials (the administrative evaluations and CARET). It is a daunting and time consuming task to do a demo in person, especially when you could walk through it on the phone. However, one of the biggest take-away lessons from the Cambridge trials is that jumping on a plane to be there in person opens up doors that would be otherwise closed.

While the demonstration approach was less frustrating for users, it sometimes had the negative impact of allowing the users to not fully engage the system.

[94] This refers to trying to train someone how to use a system by telling him the next function that he needs to perform, rather than explaining explicitly the steps required to carry out that function (where to click, etc).

25. Often you need to do the work for someone in order for him to take part in a trial use. In this case he hasn't really used the system, he has just collected results from it. (DP3, DP4)
26. When all of the work is performed for a user, there is little chance that he will use the system later on his own. (DP3, DP4)

In half of the DP trials, I ended up doing a majority of the preparatory work. In these instances, I created the surveys for the users and set up the system (and in some cases I even reported the results). These trials were useful for testing student interaction with the system, but they were worthless in testing instructor use. It would have been preferable in these trials to do less work and have the instructors do more, but the likely result would have been non-engagement by the professors, resulting in no trial at all.

The last area of learning related to instruction and new user adoption was in regards to the inevitability of user error. This was highlighted by the different levels of success that the instructors at InterSystems had with the system when they were first adapting it.

27. The slope of the learning curve can be dependent upon the user's comfort experimenting with technology. (CP1, CP2, CP4)
28. Explicit written directions concerning the use of a new system can be ineffective if the user does not pay attention to the details. (CP1)
29. As hard as you try, you can never fully remove user error from a process, unless the user doesn't interact with the system. (CP1)

There are three technical trainers that participated in the OnFORME usage trial (T1, T2, T3).

Trainer	# Surveys	# In-Class Surveys	# Post-Course Surveys	% Surveys with Difficulty	Issues
T1	3	0	3	0%	None
T2	3	2	1	33%	Minor
T3	3	3	0	100%	Major

Figure 11-3: OnFORME Implementation Records for InterSystems Trainers

Referring to Figure 11-3:

- T1
  - Had no difficulty implementing the surveys for any courses.
  - Spent time playing with the OnFORME interfaces and compiled a list of suggested improvements.
  - Due to timing and habit, ended up sending out all of his surveys after the courses were complete.

- T2
    - Had difficulty using the system during the first time it was attempted in-class.
    - Was able to resolve the issue, enabling all of the students to fill out the evaluation.
    - Had no difficulty for the next in-class evaluation, or for the post-course evaluation.
- T3
    - Was unable to use OnFORME in-class on the first two attempts.
    - On the third in-class attempt, was able to partially use OnFORME for in-class evaluations.
        - Unsuccessful in directing the students to type in the URL, T3 had them all log in with the trainer profile, and then cut and paste the URL into another browser.
        - This resulted in some students not responding, some students responding as the trainer (from within that profile), and the other students filling out the survey successfully using their own profiles.
        - This also resulted in the trainer profile being compromised, which opened the possibility of exposing all of the training and student records to the members of that course.
    - T3 believes that all of the difficulties have been sorted out, and is hoping to successfully use OnFORME during her next training assignment.

Some of the challenges encountered above were due to a design within OnFORME that did not match well with trainers' use processes. The URL that had to be typed in by the students was case sensitive, and this was not expressly communicated to the trainers. This resulted in the difficulties experienced by T2's first use, and T3's third use. It is interesting to note that one of the two trainers that encountered this issue was able to debug it on the fly (T2), while the other was unable to pin down the problem, and resorted to other tactics (T3). The design and system instruction training could be blamed (in part) for these difficulties.

However, the first two usage attempts by T3 highlighted the degree to which the responsibility of proper use rests on the user. T3 received a detailed set of instructions explaining the steps for in-class use. These instructions were very similar to those given to T2, who was able to follow the directions without any problems (the detail about the URL being case sensitive was not in the instructions). Following T3's first failed attempt, I sent another set of instructions, more detailed than the first, prior to the end of the second course. This attempt was also without success. It was not until both failed attempts that I discovered that T3 had not taken the time to carefully read my emails, and therefore did not carefully read the instructions. Their specificity and my attention to detail did no good since they were not read. Insight 29 was a particularly frustrating lesson to be reminded of. This lesson also served to reinforce earlier lessons, because if I had the opportunity to sit down with T3 and demonstrate each step of the process, then I would not have had to rely on my written instructions being read.

11.2.3 Demonstration and Training Conclusions

All of the insights that have been presented in this section support the following hypothesis:

> ❖ When instructing new users in the use of a new IT system, being there in person and teaching by demonstration increases the likelihood of adoption and use.

The above hypothesis came clearly from the experience of trying to enable the adoption of OnFORME. However, this would not apply to the adoption of all new IT systems, as there are likely exceptions to this rule. This may not be the case when one or more of the following conditions exist:

- The new users are highly technical and hands-on with IT systems
- The new IT system is a replacement for an existing IT system, and the use processes and interfaces are very similar
- The decision for adoption has already been made, and users are therefore forced to use the new IT system

In situations where the above hypothesis does apply, it is likely due to one ore more of the following situations:

- The new users are not fluent with technology experimentally
- The new users to not have the time to play with the system on their own, and do not have the outside motivation to do so
- The new system is replacing a drastically different process (be it IT based, or non-technical), and therefore the change represents a drastic shift in how things are done.

In the usage trials observed for this paper, examples of each of the above characteristics were observed.

12 Domains of Learning – Use

The previous chapter addressed learning that occurred through the adoption process of OnFORME. This chapter examines the insights gained from the use of the system through the initial usage trials. The insights in this chapter continue from where the previous chapter left off, and are marked with a light bulb ( ). Additionally, the trials in which the insights were gained are listed parenthetically, and refer to Figure 12-1.

Trials Primarily from Customer Pull
CP1 - InterSystems Classroom Training
CP2 - ESD.140 Weekly Surveys
CP3 - TP Module Evaluations
CP4 - MOTI Evaluation
CP5 - CARET Adoption
Trials Primarily from Developer Push
DP1 - ESD.801 Leadership Weekend
DP2 - ESD.801 Final Session
DP3 - ESD.80 Thesis Seminar
DP4 - 5CMI3 Evaluation
Failed Trial Attempts
FT1 - InterSystems eLearning
FT2 - Introduction to Aerospace Engineering
FT3 - TPP Curriculum Review
FT4 - CMI Sustainable Development MPhil

Figure 12-1: Categorization of OnFORME Trial Attempts

The majority of the insights gained from the use side of the trials could be summed up in two hypotheses:

- Making feedback part of the defined "role" of a student will result in a higher response rate.
- Collecting feedback from a student cohort while they are in a single location will result in a higher response rate.

The following sections outline these insights and the data that led to these hypotheses, as well as a group of insights that did not directly apply to either.

12.1 Response Rates and Incentives

One of the most important parts of requesting feedback is having people respond. Specifically, it would be helpful to know how students in academia can be convinced to participate in feedback exercises. The usage trials with OnFORME provided some interesting data points that might provide insights into enabling high response rates. The first area of insights deals with the incentives for providing feedback, and the

expectations of having to give feedback. Figure 12-2 shows response rate data from the MIT and Cambridge University trials. [95]

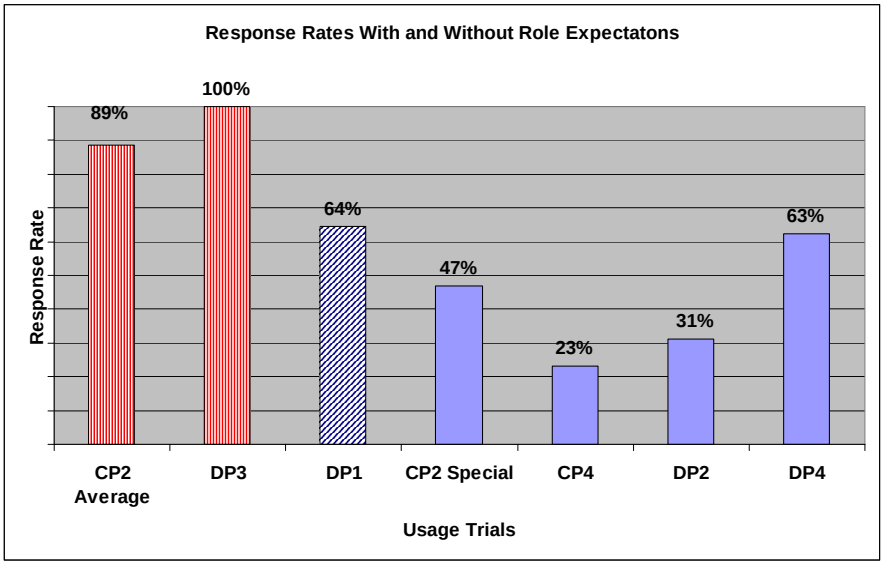


Figure 12-2: Response Rates of the Trials[96]

An explanation of the three types of bar shown in this graph follow.

- Bars with Vertical Lines (2 leftmost bars) – In these trials, giving feedback was expected as part of the students' responsibility for that class. This was communicated to them at the beginning of the semester. If students did not submit feedback, their course grade could be negatively affected.
- Solid Bars (4 rightmost bars) – In these trials, the request for feedback was a surprise to the students, and there was no incentive associated with the feedback request.
- Bar with Diagonal Lines (3rd bar from the left, 64%) – In this trial the request for feedback was a surprise, but the students were given the incentive of being shown digital pictures of the event after completion of the survey.

The trends in the graph are quite clear, in that the three trials that used an incentive had higher response rates than those without incentives. Between those that used an incentive, the trials in which the students were expecting the requests for feedback had significantly higher response rates than the feedback request that was a surprise to the students. The following insights were gleaned from this data:

[95] This first section discussing incentives focuses on the academic trials, as those were the only ones that offered incentives (in some cases). Since students in corporate training settings may have different value functions that impact their decision whether or not to give feedback, the InterSystems trials were not included in this first analysis.

[96] The "CP2 Average" statistic is the average of the 11 response rates for the ESD.140 course, and the "CP2 Special" statistic is the response rate for the experimental survey on the Sloan System Study.

30. Properly structured incentives can help increase the response rate. (CP2, DP3)
31. Framing feedback as an expectation from the beginning is an effective way of assuring responses. (CP2, DP3)

These insights, taken with the results of the trial, point toward a possible hypothesis that ties them together:

> ❖ Making feedback part of the defined "role" of a student will result in a higher response rate.

It should be noted that these trials were not intended as a systematic investigation of this hypothesis, but rather the hypothesis is being inductively postulated based on trends seen in the initial usage trials of OnFORME. There are many other factors that could have influenced the response rates. Further deductive experimentation should be done to test the validity of this hypothesis.

12.2 Response Rates and Location

The data collected from the InterSystems classroom training tests reflected patterns of a different sort. This data has been separated into two groups for analysis:

- Those surveys which were given to the students to be filled out "In-Class"
- Those surveys that were emailed to the students "Post-Course"

The response rates of these two groups are shown in Figure 12-3. The "In-Class" results are the 3 on the left, and the "Post-Course" results are the 6 on the right.

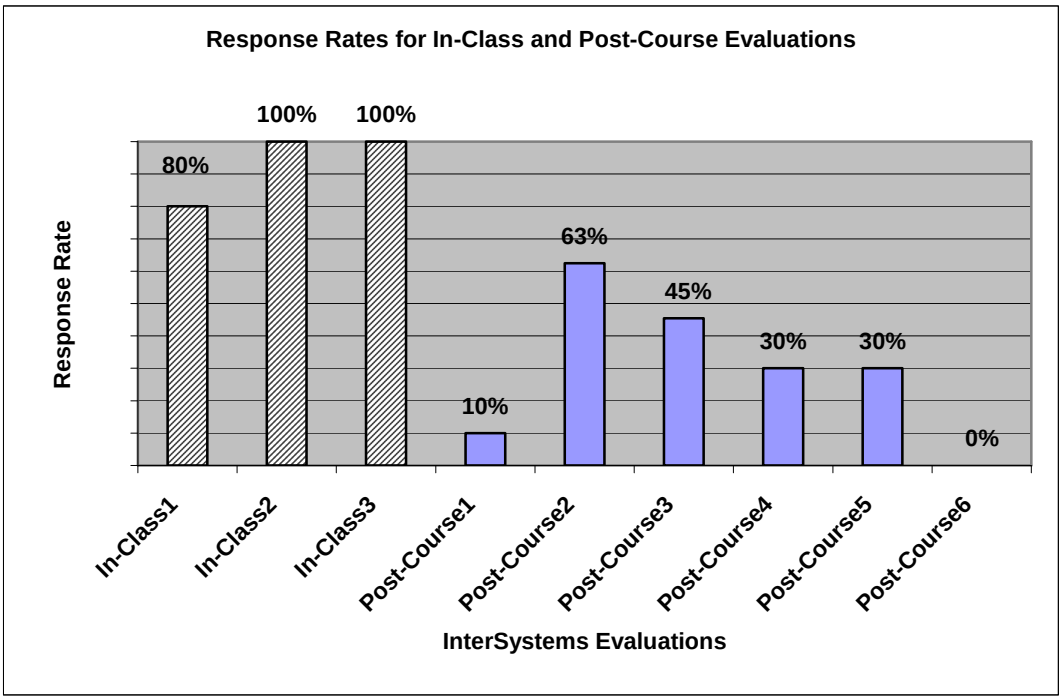


Figure 12-3: Response Rates For InterSystems Uses

There is a stark contrast in the response rates of the two categorizations of feedback requests. The three results with diagonals on the left (In-Class) were significantly higher than the six Post-Course evaluations shown on the right. The trends seen in this data resulted in the following insights:

32. When respondents are asked for feedback at the end of an event and are allowed the time and means to give it, they are more responsive. (CP1)
33. When a group shares the same feedback task, the common knowledge of that task improves peoples' willingness to respond (i.e. each person is more willing to be responsive to the survey when they understand that every other person is working on the same survey). (CP1, CP2, DP3)
34. Including feedback as part of the "schedule" prevents having to request that respondents squeeze it into their own schedule later. (CP1)

It is also instructive to consider the response rates of the Post-Course surveys in light of the time that had elapsed since the end of the course when the request for feedback went out. This information is graphed in Figure 12-4.

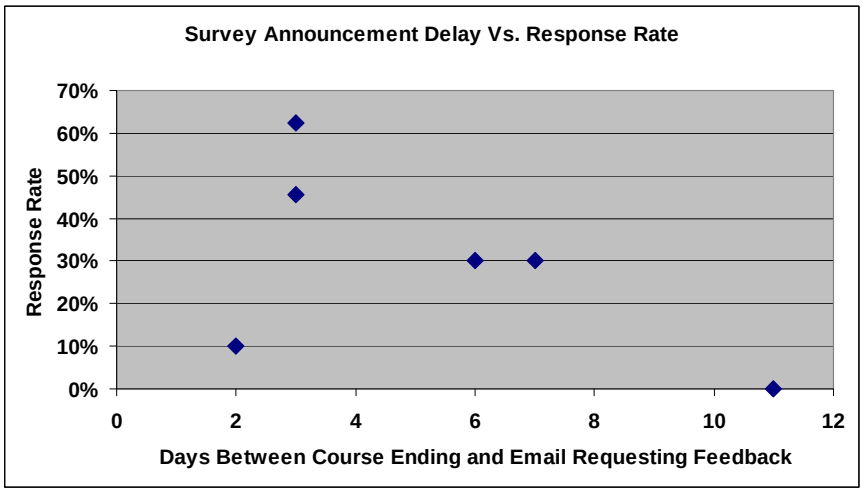


Figure 12-4: Feedback Request Delay and Survey Response Rate

If the data point on the far left were excluded as an outlier, then an obvious trend emerges in the remaining points. This trend is downward and slopes to the right, indicating:

35. As the time increases between the event and the feedback request, the likelihood of receiving a response decreases. (CP1)

It is interesting that the above trend appears to be so pronounced even though the longest delay is only a week and a half.

Again, given the small number of data points and the lack of uniformity between the different feedback requests, these insights are speculative at best. However, the insights and trial data can be combined to posture the following inductive hypothesis:

- ❖ Collecting feedback from student cohorts while they are in a single location will result in a higher response rate.

It would be interesting to conduct a further deductive investigation to see to what extent the above holds true, under what circumstances it holds true, and what other factors can influence its effectiveness.

12.3 Additional Insights

Beyond the use insights included in the previous two sections, there were several other nuggets of learning that came from the use trials and are worth mentioning. These insights are listed below, with comments clarifying their context.

36. Sending a feedback request at the end of a semester to students with no incentive to respond will likely produce a low response rate. (DP2)
    a. The survey sent to the ESD.801 students at the end of the semester did not elicit a very high response rate. Due to the fact that many students did not remember receiving the email about the survey, I think it is safe to conclude that trying to collect survey responses when students are stressed by the demands of the end of the semester will yield a low response rate. This can also be seen in the ESD.140 response rates that dipped towards the end of the semester.

37. Delegating content entry responsibility to the end user can reduce the administrative overhead of populating a system with user information. (CP1, CP2, DP3)
    a. The first beta test highlighted the challenge of an instructor having to create a class list within OnFORME. In the absence of integrating OnFORME with a complete CMS, the next best alternative is to allow students to "Self-Register," which allows the students to do the data entry work for themselves. This process was used for most of the remaining trails, and it resulted in cleaner use tests from the perspective of the instructors.

38. Users are much more patient with new systems when they understand that it is a student project, than they would be under other circumstances. (CP1, DP1, DP4)
    a. This general observation was made from most of my interactions with professors, who were very supportive of this application and research effort.

39. Information pushed to the users is more useful than information waiting for retrieval. (CP2)
    a. Dr. Cutcher-Gershenfeld explained that he viewed the students' weekly feedback less frequently then he had planned – approximately every 2-3 weeks rather than weekly. This was due to his having to set aside time to log in to OnFORME and read the responses. He said that had the information been sent to him in a weekly update, he would have been more likely to review it. The information would have been much more useful because of its timely delivery.
40. Some students have a powerful fear of "big brother," and the thought that a feedback system could correlate their responses over time terrifies them. (DP3)
    a. During a Q&A presentation of the OnFORME system for the TPP Thesis seminar, I was surprised by the reservations voiced by the students concerning the vision of OnFORME being used broadly to collect feedback information.[97] Some of the students disliked the idea of having a system that collected surveys for many classes, and then had all of the responses linked to personal profiles. They were afraid that somehow that data might be used against them. This highlighted possible resistance to this paradigm of ubiquitous feedback in education, which would have to be mitigated if this system were to ever be used widely.

Altogether, the use trials of the OnFORME system highlighted many useful insights, and helped form several hypotheses that could be thoroughly investigated by future researchers.

[97] It should be noted that the students did not have concerns with using the tool to collect feedback in that course. Rather they were objecting to the principle of having a single system be used in all of their classes during their several years at MIT. They are afraid that the aggregate of their responses might somehow be twisted and used against them.

Part IV – Conclusions

13 Conclusions

The OnFORME application development process represented a large amount of learning on both the technical and social use sides, and produced some novel initiatives in this application space as well as some adoption and use hypotheses to be further investigated.

13.1 Technical Implementation Findings

The highlights of the technical output from this project include:

- A robust data model for educational structures in relation to feedback collection
- A flexible data model for reusable feedback objects and their subparts
- Extensive Feedback Object reuse enabled by the separation of Data from Presentation properties within the architecture.
- An implementation of "context adaptive" Instructor Specific questions, which become tailored to represent the instructors teaching the course in which the questions are used.

As discussed in this thesis, the combination of the above strengths, in addition to other minor novel features, has resulted in an application that contains a notable amount of non-trivial innovation when compared to other applications in this space.

In addition, other researchers or users can now leverage the technical accomplishments as OnFORME has been distributed under the Open Source "MIT License". The code, documentation, and this thesis have been made available at http://onforme.sourceforge.net.

13.2 Social Implementation Findings

There were 40 different discrete insights gleaned from the adoption and usage trials of OnFORME. From these insights and the data collected during the trials, I was able to inductively generate the following hypotheses:

Concerning Adoption of a new Web based Feedback System

- ❖ User pull is strongest when the new system is an improved replacement for an existing process, and that process is well understood.
- ❖ When instructing new users in the use of a new IT system, being there in person and teaching by demonstration increases the likelihood of adoption and use.

Concerning Use of a new Web based Feedback System

- ❖ Making feedback part of the defined "role" of a student will result in a higher response rate.
- ❖ Collecting feedback from a student cohort while they are in a single location will result in a higher response rate.

These hypotheses are put forth for future investigation in deductive research through controlled experimentation.

The sum of the social use learning could be encapsulated in the following generality:

Successful adoption and use of a new web application is a function of understanding and adapting to the organization's:

- process
- strategy
- organizational structure

Additionally, the new application must be solid technology that constitutes an improvement over the present state.

This general observation maps to the above hypotheses in the following ways:

- With an understanding of existing organizational processes and structures, there is an increased chance of finding users most likely to generate pull, and therefore the chances of adoption rise.
- Visiting a potential user for a face-to-face demonstration and instruction greatly facilitates one's understanding of the organization, thus facilitating identification and choice of target users most likely to adopt the system.
- With an understanding of the organization's current structure, it is easier to identify the channels and individuals with the ability to adopt feedback as part of students' defined role.
- Similarly, the processes of an organization can be analyzed to find ways in which feedback can be collected when the students are collocated, thus improving use effectiveness through increased response rates.

To summarize, the major lessons showed that adoption is more likely to take place when the new system is a drop-in replacement for an old process. The value of the feedback system is directly related to the response rates achieved, which can be improved through role-expectation management and collocation of the population during feedback.

14 Future Work

To date, the OnFORME system represents a vertical application of moderate functionality. However, given the scope of the need, there are many ways in which the application could be improved. The following sections describe briefly the vision for some of these improvements. Many of these additional features have been foreseen and are already included in the data structures, but need functional implementation.

14.1 Differentiating "Levels" of Instructor Assignments

One of the central innovations of OnFORME is its ability to make questions instructor-specific such that they adapt to the course context in which they are used. Additionally they partition the results in such a way to preserve confidentiality of personal critiques of instructors. This innovation can be extended in a powerful, yet rather simple manner by adding a teaching hierarchy within courses, and then allowing questions to be focused on points of the hierarchy. For example, suppose an institute had a standard survey that they gave at the end of the semester, which collects feedback on the teaching abilities of the instructor and TAs for that course. Under the currently implementation, if these questions are labeled as instructor-specific, then the same questions would be asked about both the instructor and the TAs. But, if those involved with the teaching of a course could be differentiated based on the type of teaching role they held, then it would be possible to have questions that triggered off of that differentiation.

The feature would be rather simple to implement. First, a `Level` property should be added to the `TeachingAssignment` class. This property could either be an integer ranking (1, 2, 3, etc) or some sort of string label ranking ("primary", "secondary", "tertiary", etc). It would be necessary to create some way of accessing this field when an instructor is added to a course. Next, the `QSlot` class would need to be changed to provide a way to record the instructor level (or levels) to which its `Question` object is meant to apply. This `Level` property also needs to be given an interface through which it can be set. Finally, the `RenderHTML()` method in the `QSlot` class would need to be altered to implement this change, by only rendering instructor-specific questions for those instructors which have been assigned that level for that course.

This is a rather simple concept, but powerful when coupled with survey reuse and context-adaptability, thus minimizing the time required to create and adapt surveys by hand for different courses.

14.2 Author Specific Feedback

Feedback to material authors is a central necessity for the improvements of learning materials, however there is currently no way to automate this process. OnFORME was created with the future vision of enabling automated feedback to material authors (although it was not a high enough priority to have it included in this iteration of the software). Finding a way to incorporate this functionality into this system would be a substantial improvement to the value of the system.

14.2.1 Feedback for Authors of Subjects

Originally, the educational structures portion of OnFORME was designed with the goal of author-feedback in mind. The architecture was drafted to associate authors with subjects, for the purpose of future feedback functionality. The `AuthorCredit` class is a many-to-many associative class that connects people in the database with subjects that they are credited for authoring. This provides a rather simple means of identifying the recipients of feedback on the course materials for a subject.

To implement this feedback loop, it would be necessary to first implement an interface through which people can be assigned to subjects as authors (to date, this has not been completed). Next, a property should be created within the `QSlot` class, which classifies the `Question` object in that slot as "Feedback to Subject Author" (this can be accomplished with a property similar to the `InstructorSpecific` property that already exists). When this flag is set, there would be no change in the rendering of the slot (`RenderHTML()`), however the code that receives the submitted responses and adds them to the database would have to be altered. This code must be changed to compile the questions and answers marked as feedback to the authors, and email them to the list of people recognized as the authors for that particular `Subject` object (in which the `Survey` object is used). Email is preferable, because the authors of subjects will likely not frequent the OnFORME system to check for submissions labeled for them. However, it would be helpful to create a portal for authors, through which they could view all feedback that has been collected for their subjects.

14.2.2 Feedback for Authors of Learning Objects

The current architecture will support feedback to authors of entire subjects, however a considerable amount of work will need to occur (including additions to the architecture) in order to enable automated feedback to authors of learning objects. A learning object is a discrete package of material, which is intended to teach a certain number of learning objectives. Usually, instructors will include multiple learning objects in a single course. Another way to phrase this is to say that many authors contribute pieces of a curriculum. It is therefore not an adequate solution to assign as authors everyone who contributed any portion of a subject (as described in the last section), because they would receive feedback on materials to which they did not contribute. If OnFORME is going to accommodate automated feedback to the authors of learning objects used in subjects, then the architecture will need to be changed. The final functionality should include some means of identifying authors of portions of a subject (or a course, because learning objects might be used by one professor but not another in the same subject). Additionally, there needs to be some means of tagging questions with pointers to learning objects, so that the responses are marked and routed to the authors. This may involve substantial changes to the system, but will provide a large amount of additional value to authors for improving their learning materials.

14.3 Horizontal Feedback Between Peers

The current feedback paradigm in OnFORME is vertical feedback from students to instructors. It would be a very useful enhancement to create mechanisms that enable

horizontal feedback. This could be students evaluating each other (e.g. for class presentations), or authors engaging in peer review of learning material. This type of feedback is possible at this point, but only by manipulating the structures (e.g. creating a dummy course to hold peer review feedback). Adding mechanisms that natively allow horizontal feedback would make OnFORME more user friendly in a variety of different domains of use.

14.4 Editable Respondent Submissions

A simple but useful extension to the system would involve implementing the functionality that would allow respondents to be able to edit previous survey submissions. This functionality is already built into the data model as part of the `SubmissionMode` property of the `FeedbackRequest` class. Additionally, the control of that variable is included in the `EditSurveyUse.csp` page, but it is currently commented out.

In order to implement this feature, changes should be made to the `ShowSurvey.csp` page, which would populate a survey with the answers already in the database (if they exist). This could be done with a function that writes JavaScript after the survey has been rendered, or this could be included as part of the `RenderHTML()` method, where the answers are populated as the survey is rendered. Probably the former solution would be simpler. A later point of feature expansion would be to implement the ability to edit multiple responses, but that would require an additional interface page for the respondent (in choosing between existing responses in the database).

14.5 Graded Feedback Objects

The initial implementation of OnFORME focused on use-cases for non-graded surveys and polls. However, the feedback engine could easily handle graded quizzes with the addition of a moderate amount of functionality. There are two sets of functionality that could be implemented: the introduction of "correct" answers; and giving feedback to the students concerning the "correct" answers.

14.5.1 Implementation of "Correct" Answer Options

The first step in grading assessments would be to implement the ability to designate the "correct" `AnswerOption` object(s) contained in the composite question. This pointer is already contained in the data structure, and it would simply require that the interface to the question builder (`SlotProperties.csp`) be adapted to enable the correct answers to be flagged (alternatively, another approach would be to include a "correct" property in the `MCAnswer` class – both approaches have pros and cons). Following this implementation being added to the building of the survey, the next step would be to change the reporting pages to indicate correct answers and the percentage of students who responded correctly, etc.

14.5.2 Implementation of Feedback to Students

Instant feedback to students is a useful teaching tool, and it is one that is often used in online quiz tools. This functionality would allow instructors to choose to show the

"correct" answer to the student following the submission of a response. This would be slightly more challenging to implement, as additional student interfaces would need to be constructed (or the current ones changed) to support this functionality. One approach might be to redirect students to a new page that shows the survey, their answers, and notation indicating correct or incorrect responses (with the correct answer marked). This would be a substantial add-on to the system, and user analysis would be wise to determine the existence of demand, prior to taking the time to implement this.

14.6 Multi-Page Feedback Objects

One fairly common feature in assessment engines that has not yet been implemented in OnFORME is the creation of multi-page surveys. This feature allows assessments to be broken into smaller chunks of questions that the respondent navigates through. The concept of multi-page assessments opens up a range of additional features that might be considered for implementation in tandem. These features include:

- Options for choosing which questions go on which pages.
- The creation of a "progress meter" which shows the respondent how many pages or questions they have left in the survey.
- The option for allowing surveys to be started at one point, and then finished later (this aligns with the concept of the "editable" survey feature, discussed in 14.3).
- The ability to allow the user to go backwards in a survey to change previous answers.
- The ability to prevent users from going backwards in a survey to change previous answers (e.g. for quizzes).
- The ability to create adaptive question branching, in which survey authors can choose alternative paths through a survey based on the responses to certain questions.

The basic structure for multi-page surveys is included in the data-model (i.e. the `Page` property in the `Slot` class), but there would be a moderate to large amount of additional coding required to implement this functionality, depending on which of the above features are included.

14.7 Rules-Based Email Triggers

Automating the follow-up communications process would provide for a large amount of time savings for the instructors, and add significant value to the OnFORME system. The use case would be for instructors that would like an email automatically sent to all students in a course that have not responded to a certain survey by a certain date, or alternatively having an email sent to those who have filled out a survey. This functionality was foreseen in the architecting stage of the application, but it has not yet been implemented. Future development aimed at implementing automated email should consider the `EmailTrigger` class, which was created with the intent of implementing this functionality. However, the implementation processes was not thoroughly thought through, and therefore changes to that class (and perhaps to other classes) might be necessary. Additionally, the creation of the interfaces and the underlying code would be required.

14.8 Import / Export of QTI Compliant Feedback Objects

As mentioned earlier, QTI is an IMS standard for defining assessment objects in XML. If this standard becomes widely adopted, it may become helpful to create various levels of QTI functionality into OnFORME. The first step would be to implement a QTI export feature, which would allow OnFORME created assessments to be used by other QTI-compliant assessment systems (which have import functionality). The next step would be to implement an import feature, which would allow QTI assessments created on other systems to be used in OnFORME. The final step would be (if there were a perceived need) to create a web service that would allow QTI surveys to be sent to or received from other systems. This last step is less likely to be necessary, but as it is conceivable that the need might arise in the future, it was listed as an option to consider.

14.9 Additional Reports / Analysis Tools

At present, OnFORME comes equipped with a functional but modest set of reporting interfaces. The three current options are:

- viewing the summary of a feedback request response set
- viewing individual responses
- viewing a matrix with all response set data, which can be cut and pasted into Excel for further analysis

Due to the current limited reporting functionality, there is a wide range of additional reporting features that can be added to the system. Some of these features include:

- The ability to export a result set into a CSV (comma separated values) file, so that it can be easily imported into Excel or any statistical analysis program.
- The ability to create an executive summary of all responses given for a survey object.
- The ability to select the uses of a survey which are to be included in a result set for analysis.
- The ability to view the aggregated responses for particular question objects (where the question might be used in several surveys, across several feedback requests).
- The ability to run advanced statistical analysis on a result set (currently, percentages are the only numbers calculated on the aggregate set).

Due to the fact that all of the data is in the database, collecting that data which is to be aggregated, displayed, and analyzed should not be a difficult problem for someone knowledgeable in SQL. Any of these reporting functional improvements can be created individually, with additional features being added as the needs for them arise.

14.10 Integration with Other Course Management Components

As mentioned in the beginning of this thesis, due to time and resource constraints, OnFORME was designed from the beginning to function as a standalone vertical application. However, this does not preclude further development from enabling it to work with other applications or systems within educational environments. The functionality of OnFORME can be extended to other systems through the use of Web

Services without too much difficulty, as Caché includes native Web Services classes that allow any class to be exposed and projected as a web service.

The most difficult part of integrating the OnFORME system with other applications would be deciding which information needs to be passed, and the structure of that information interchange. This could be done on a case-by-case basis (based on the specifics of the applications with which OnFORME is to be integrated), or by adapting an educational application open standard, such as OKI or Sakai. The decision of which path to follow, and the points of the architecture to expose, will be a matter of preference based on the situation in which integration proves to be helpful (or necessary).

One very powerful integration upgrade might be integrating OnFORME with a course management system (CMS), so that the course structures referenced for course lists (and instructors) are retrieved from the CMS. This could occur either through a batch process, or in real time. This would automate most of the educational structures side of the system, allowing OnFORME to be used solely for its survey engine (which is the central value-add of using it at all).

However, while integration with other systems is certainly possible, it may require a large amount of work, and the benefits should be carefully weighed prior to embarking on such a major rework of the OnFORME architecture. Until integration proves to be the most valuable path, OnFORME can continue to function as a vertical application.

14.11 Future Vision and Concluding Statements

While there remain many ways in which OnFORME can be improved, the progress, learning, and achievement to date represent significant advances towards the vision of ubiquitous educational feedback. The initial usage trials described in this thesis highlighted a number of reasons why this type of application is valuable, including significant reduction in administrative tasks, improved understanding of ways to increase response rates, and a more rapid sampling of students' lecture comprehension. Additionally, by publishing the OnFORME code base with an open source license, it is entirely possible for future researchers to pick up from where this work left off in the technical, as well as social arenas of learning. However, this work has also shown that this application (or others like it) require the right combination of existing processes and personalized instruction in order to induce adoption.

There is a tremendous amount of untapped knowledge in the form of student opinion and observation that is just waiting to be discovered in classrooms across the globe. Perhaps through the continuation of work on OnFORME, and systems similar to it, it will be possible to extract more of that knowledge by eliminating the process barriers that currently stand in the way of ubiquitous feedback. There is yet a long way to go, but the goal is not impossible, given the proper application of appropriate technology, tempered by a thorough understanding of the processes and players upon which that technology will depend for adoption and use.

15 References

Educational Assessment Products

Blackboard, Cambridge University Installation, http://bb.caret.cam.ac.uk, (last accessed April 5th, 2004).

M. Brown, "QTools – A Web Survey Tool for MIT," Presentation, *MIT IAP 2004*, Jan 15th, 2004.

M. Brown, AMPS Web Development, MIT, (Personal Interview), *QTools*, April 2nd, 2004.

R. Espinosa, Systems Research Programmer, University of Michigan, (Personal Interview), April 9th, 2004.

Fast, http://www.getfast.ca, (last accessed April 5th, 2004).

Flashlight Online, http://www.tltgroup.org/programs/flashlightonline.html, (last accessed April 5th, 2004).

T. Henderson, Assessment Coordinator, Center for Teaching, Learning, and Technology, Washington State University, (Personal Interview), *Flashlight Online*, April 13th, 2004.

C. Kerns, Manager of Information Technology, Academic Computing, Stanford University, (Personal Interview), April 13th, 2004.

V. Krist, Software Architect, Pearson Education, (Personal Interview), *Quizlab*, April 10th, 2004.

Navigo, http://navigo.sourceforge.net/, (last accessed April 5th, 2004).

Z. Patz, FAST Architect, Mount Royal Collage, (Email Interview), *FAST*, April 2nd, 2004.

QTools, http://amps-tools.mit.edu/qtools/, (last accessed April 5th, 2004).

Quizlab, http://www.quizlab.com, (last accessed April 2nd, 2004).

S. Saltzberg, Senior Consultant The TLT Group, (Personal Interview), *Flashlight Online* April 9th, 2004.

Sloanspace, http://sloanspace.mit.edu/dotlrn, (last accessed April 5th, 2004).

L. Speelmon, Principal Systems Analyst , Indiana University, (Personal Interview), *Navigo*, April 2nd, 2004.

L. Speelmon, Principal Systems Analyst , Indiana University, (Personal Interview), *Navigo*, April 4th, 2004.

E. Walker, CEO IMS Global Learning Consortium, Inc., (Personal Interview), April 9th, 2004.

Programming Methodology

K. Beck et. al., *Manifesto for Agile Software Development*, http://www.agilemanifesto.org/, Feb. 2001, (accessed April 4th, 2004).

B. Boehm, "Get Ready for Agile Methods, with Care," *Computer*, Jan. 2002, pp. 64-69.

D. Box, "'Indigo': Services and the Future of Distributed Applications," *Microsoft Professional Developers Conference*, Oct 26-30 2003.

M. Fowler, *The New Methodology*, http://www.martinfowler.com/articles/newMethodology.html, Rev. April 2003, (accessed April 4th, 2004).

E. Gamma et. al., *Design Patterns: Elements of Reusable Object-Oriented Software*,, Boston: Addisin-Wesley, 1995.

O. Gazitt, "'Indigo': Web Services Protocols And Architecture," *Microsoft Professional Developers Conference*, Oct 26-30 2003

F. Maurer and S. Martel, "Extreme Programming: Rapid Development for Web-Based Applications," *IEEE Internet Computing*, Jan - Feb 2002, pp. 86-90.

M. Weisfeld, *The Object-Oriented Thought Process*, SAMS, 2000

D. Wells, *Extreme Programming: A Gentle Introduction*, http://www.extremeprogramming.org/index.html (accessed April 4th, 2004).

D. Wells, *XP and Databases*, http://www.extremeprogramming.org/stories/testdb.html (accessed April 5th, 2004).

"Why Adopt the Caddie.Net Framework and Web Services," http://iesl.mit.edu/web/default.aspx, (accessed April 7th, 2004).

Web-Based Assessment

K. Hmieleski and M. Champagne, "Plugging in to Course Evaluation," *The Technology Source*, Sept/Oct 2000.

"IMS Question & Test Interoperability Specification", *IMS Global Learning Consortium, Inc.* http://www.imsproject.org/question/index.cfm, (accessed April 7, 2004).

IMS Question & Test Interoperability Specification: A Review, Document: IMSWP-1 Version A, 10th October, 2000.

X. Liang and K. Creasy, "Classroom Assessment in Web-Based Instructional Environment: Instructor's Experience," *Practical Assessment, Research and Evaluation*, March 2nd, 2004, http://PAREonline.net/getvn.asp?v=9&n=7, (retrieved March 31st, 2004).

C. Mertler, "Patterns of Response and Nonresponse from Teachers to Traditional and Web Surveys," *Practical Assessment, Research and Evaluation*, Oct 7th, 2003, http://PAREonline.net/getvn.asp?v=8&n=22, (retrieved March 31st, 2004).

B. Ravelli, "Anonymous Online Teaching Assessments: Preliminary Findings," *ERIC*, July 15th 2000, http://ericae.net/ericdc/ED445069.htm.

B. Ravelli, "The Potential and Challenges of Anonymous Online Student Evaluation of Instruction," *ADETA: The Distance*, Nov 2002, Volume 11, Issue 3, pp. 1, 6-7.

D. Solomon, "Conducting Web-based Surveys," *Practical Assessment, Research and Evaluation*, Aug 23th, 2001, http://PAREonline.net/getvn.asp?v=7&n=19, (retrieved March 31st, 2004).

User Interviews

J. Ash, Professor, Cambridge University, (Personal Interview), March 16th, 2004.

M. Ben-Ur, Learning Services, InterSystems Corp., (Personal Interview), June 20th, 2003.

J. Breen, Director of Learning Services, InterSystems Corp., (Personal Interview), June 20th, 2003.

J. Breen, Director of Learning Services, InterSystems Corp., (Personal Interview), March, 2004.

H. Cruickshank and F. Roques, Sustainable Development, Cambridge University, (Personal Interview), March 16th, 2004.

J. Cutcher-Gershenfeld, Sr. Research Scientist, Sloan, (Usage Test), December, 2003.

J. Cutcher-Gershenfeld, Sr. Research Scientist, Sloan, (Usage Test), February, 2003.

A. Dane and J. Cutcher-Gershenfeld, ESLC, (Personal Interview), June 20th, 2003.

E. Dogbegah and S. Beelaerts, TP Student Representatives, Cambridge University, (Personal Interview), March 15th, 2004.

E. Dogbegah and S. Beelaerts, TP Student Representatives, Cambridge University, (Personal Interview), March 17th, 2004.

O. deWeck, Professor, Department of Aeronautics & Astronautics, MIT, (Person Interview), May 22nd, 2003.

D. Hackett, Sales Engineer, InterSystems Corp., (Personal Interview), June 3rd, 2003.

J. Jensen, WebMaster/New Media Specialist, InterSystems Corporation (Personal Interview), March 11th, 2003.

R. Kirchain, Assistant Professor, MSE & ESD, MIT (Usage Test and Personal Interview), February 19th, 2004.

M. Mandolis, Administrative Assistant, Technology & Policy Program, MIT, (Usage Test), October, 2003.

A. McDaniel, Technical Trainer, InterSystems Corp., (Personal Interview), June 20th, 2003.

S. Miller, Course Administrator, Technology & Policy Program, MIT, (Personal Interview), September, 2003.

B. Nuttall, Course Director, MPhil in Technology Policy, Cambridge University, (Personal Interview), Dec 8th, 2003.

B. Nuttall, Course Director, MPhil in Technology Policy, Cambridge University, (Personal Interview), March 15th, 2004.

B. Nuttall, Course Director, MPhil in Technology Policy, Cambridge University, (Personal Interview), March 16th, 2004.

S. O'Leary, Technical Trainer, InterSystems Corp., (Personal Interview), June 20th, 2003.

S. O'Leary, Technical Trainer, InterSystems Corp., (Personal Interview), March, 2004.

N. Oliver, Professor, Judge Institute of Management, Cambridge University, (Personal Interview), April 15th, 2004.

A. Shokrollahi, (Personal Interview), June 24th, 2003.

A. Shuman, Administrative Assistant, InterSystems Corp., (Personal Interview), March, 2004.

A. Shuman, Administrative Assistant, InterSystems Corp., (Usage Test), March, 2004.

J. Solon, Sr. Technical Trainer, InterSystems Corp., (Personal Interview), June 20th, 2003.

J. Solon, Sr. Technical Trainer, InterSystems Corp., (Personal Interview), March, 2004.

J. Solon, Sr. Technical Trainer, InterSystems Corp., (Usage Test), March, 2004.

P. Sparling, Course Administrator, MPhil in Technology Policy, Cambridge University, (Personal Interview), March 17th, 2004.

R. Williams, Faculty Liaison, CARET, (Personal Interview), March 17th, 2004.

Other

T. Angelo and K. Cross, *Classroom Assessment Techniques: A Handbook for College Teachers*, 2nd Ed, San Francisco: Jossey-Bass Publishers, 1993.

J. Cutcher-Gershenfeld, Sr. Research Scientist, Sloan, (Personal Discussion on Methods), May 12, 2004.

W. Foote Whyte (ed.), *Participatory Action Research*, Newbury Park: Sage Publications, 1991.

Appendices

Appendix A – Use Case Chronology

Week Start Date	Social Inputs / Use Cases	Source
1-Jun-03		
	It is important to consider the sampling rate of the feedback. As an author, I wouldn't want feedback being sampled every week on a system study if I'll only update it once a semester	DeWeck
	Latency of feedback loop is also a consideration. There is value to be had from near-zero latency systems, in which students can be surveys in real time (in class) anonymously about learning, and the results are available to the instructor who can adjust the lecture pace accordingly	DeWeck
	One of the biggest challenges of eLearning is that it is difficult to read the students and get feedback on their understanding.	Hacket
	Training is an important requirement for feedback tools, as the instructor may not be familiar with the particular tool	Hacket
15-Jun-03		
	It would be useful to have a channel through which one peer instructor could give feedback to another instructor concerning learning materials	Dane & C.G.
	Students should be able to give feedback to instructors on learning materials in use	Dane & C.G.
	It would be very helpful if students could give feedback which gets channeled back to the authors of the learning material	Dane & C.G.
	This could be used to create a channel through which peer feedback can be sent to material author	Dane & C.G.
	Allowing users of learning objects to give feedback to the publishing organization of that material would be a useful feature	Dane & C.G.
	Use of use feedback data from a learning object user (instructor) to the publishing organization would be helpful	Dane & C.G.
	There is a need to capture feedback from some prior to an event (use of a learning object, a class, etc).	Dane & C.G.
	There is a need to capture feedback directly following an event .	Dane & C.G.
	There is a need to capture feedback at some designated amount of delay (months or years) following an event.	Dane & C.G.
	Means to connect a person with a feedback request might include - email (embedded in an email???) - email pointing to a website - collecting it with hardcopy, and entering it - web based	Dane & C.G.

	The content of an evaluation should be able handle: - feedback on material format, presentation - feedback on content knowledge associated with materials - feedback on demographics of person filling it out - feedback on perceived capability before and after a session	Dane & C.G.
	"People are tired of being treated as databases" - how can this be worked around?	McDaniels
	what would really be helpful would be to contact the students a month after class to find out: - what they would have like to have covered that was not, - what was particularly useful - what was the actual value of the class	McDaniels
	In the corporate setting, there is no need for personal info (except to follow up with unhappy customers), but there is no real reason why anyone would want to only answer anonymously	McDaniels
	In a course feedback, people need to be given the option of having the instructor follow up.	McDaniels
	After every course, we would like to be able to do an assessment of the course content and the instructor.	Breen
	It would be very useful to send an email out a month later, with a follow-up survey asking people where the gaps were and what the most useful parts of the course were	Breen
	There is currently a set of browser based, online tutorials which we ship to customers, but have no idea of how useful they are. It would be very useful to have surveys intermittent in those tutorials which test the knowledge of the users (and tells them if they got the question right or wrong), and that will let us know what they are learning, how many people are using them, etc. There could also be a usefulness survey at the very end.	Breen
	It is very important that any feedback collection tool have very good reports at the end of the process.	Breen
	Course Instructors would benefit from gaining immediate feedback and ratings on their performance.	Breen
	The performance ratings given by the students should have tabulated averages, so that they are easy to use on performance appraisals.	Breen
	Confidentiality of instructor ratings is very important - they should only be viewable by the instructor, their manager, and administrators	Breen
	It is important that the survey responses are secure, and that they cannot be changed by the instructors.	Breen
	Is it possible to use an online feedback system as some sort of a class continuation? Create a community of learning?	Solon
	A feedback and course organization system would be helpful if there were a way for instructors to add notes about the course, so when students contact them later on, they can see their notes and remember specifics about that circumstance	Solon

	It would be useful if there was a way to take notes on students, and create some sort of information knowledge ranking after class.	Solon
	There are many ways in which this could be implemented in the eLearning context: - in class surveys - post class surveys - evaluation after 1 free class - evaluation 1 months after someone purchases a subscription - evaluation 3 months after someone purchases a subscription	Ben-Ur
	It would be important that the system maintain information about eLearning subscriptions, such as their username and password, their subscription duration, and when it began and when it ends	Ben-Ur
	It would be nice if this were set up to be a post-class question portal.	Ben-Ur
	Time delayed surveying would be most helpful, as it would help give a better understanding of what people actually learned, as opposed to what they think they learned.	O'Leary
	The feedback tool would be helpful in clarifying for her understanding what things might have been unclear or incomplete	O'Leary
	Is it possible to get instant content based feedback while the class is in the middle of a lecture? To test their understanding of the material??	O'Leary
22-Jun-03		
	to make this versatile, then the questions need to be classified according to type (true/false, etc).	Shokrollahi
	It will be less hassle on the respondent if their personal information is called up automatically by use of a cookie	Shokrollahi
	On the corporate side, it would be most useful if the system could pull company information from the global customer database using web services	Shokrollahi
	It is important that student information can be validated by a company administrator, but it would helpful if it could initially be entered by the student. Possibly there should be a dummy company field that can be validated?	Shokrollahi
	For internal login, it would be helpful if user lookup and authentication were done using web services with the Employee database.	Shokrollahi
	There is a need for extra fields to be contained in the Address object such that international users will not have a problem using this system	Shokrollahi
	It is important that when a class is canceled, the system responds in the most appropriate fashion. E.g. Sending an automated cancellation email, and removing the survey from use, etc.	Shokrollahi
28-Sep-03		

	There needs to be a way to control the number of columns displayed in text boxes	Manolis
	There needs to be an explanation on how to close the preview window	Manolis
	There is a need to be able to have a text input without a radio button (e.g. "please explain")	Manolis
	The current preview is confusing, can it be a different color?	Manolis
	There needs to be a general description on how surveys are created in the system	Manolis
	It needs to be easier to figure out how to change a login password.	Manolis
	It is not clear how to change the name of a survey, there needs be a clearer indication.	Manolis
	It is not clear how to resuse question text	Manolis
	The interface is slightly confusing in that the preview does not update unless an additional place is clicked.	Manolis
	The pop-up windows tend to fill up the screen, is there a way to prevent this?	Manolis
	Edit Person is confusing, because there is no apparent place to view what you've just entered, and no indication that there is any kind of new record made. You know you hit the save button and it saved, but when doing it repeatedly, it can get confusing	Manolis
	Couldn't figure out how to see people that had already been entered	Manolis
	There needs to be some sort of a document that explains how the different pieces fit together.	Manolis
	It the system would be much more usable if there were a way to import a large number of people into the system	Manolis
	A very helpful feature would be a way in which things could be exported from the system	Manolis
	Make it possible to batch input course lists	Manolis
	Add Student ID Number to person class	Manolis
5-Oct-03		
	It should be possible to preview a survey before you save it.	Barrett
	change Slot properties popup window from "Close" button "Continue", because it makes more sense	Barrett
12-Oct-03		
	add a note about Java being required to do the survey	Lim
19-Oct-03		

	Allow Anonymous Login & Survey completion	
	Create a "Locked" field for questions and labels so that things won't be changed accidentally	Joel C.G.
	It is not clear what is meant by "Use" as Survey. Create a warning with "Use" which explains that any changes made to the questions will effect every place they are used	Joel C.G.
	Implement automated emailing	
	verify that person is member of Course list	
	Let the User Know that they have the option to respond anonymously - build a "Respondent Info-Bar"	
23-Nov-03		
	It would be helpful to have a function that automatically created rank ordering questions	Joel C.G.
	"Label" is confusing, should it be changes to "Block of Text"?	Joel C.G.
	"Change Name" in Survey Name is confusing, can it be changed to "Enter"?	Joel C.G.
	"Magic Quotes" in cut/paste from Microsoft Word gives strange characters, can this be prevented?	Joel C.G.
	The Text Entry box needs to be larger for inputting text	Joel C.G.
	Having to add labels is not preferable, can this be changed?	Joel C.G.
	Looking at a survey overview, there needs to be more information on the questions themselves.	Joel C.G.
	It would be very helpful to have a style sheet effect the entire survey, might it be possible to integrate that functionality?	Joel C.G.
	There needs to be a system which prevents two people from editing the same survey at once.	Joel C.G.
	There needs to be a note stating that "Use" is different from "Content" e.g. question style	Joel C.G.
	it would be useful if questions could be indented	Joel C.G.
	Is there any way of making a ranking question?	Joel C.G.
	Documentation is necessary explaining the difference between Radio and Option questions.	Joel C.G.
	Can there be a way of communicating the difference between radio and option buttons?	Joel C.G.
	Most questions will probably use vertical answer options, so that should be the default format.	Joel C.G.
	The means of finding former questions is confusing and needs to be reworked	Joel C.G.

	Is it possible to select prior use statistics?	Joel C.G.
	With Radio, include "Select 1" option, defaulting to off	Joel C.G.
	Can there be Universal Options for survey creation?	Joel C.G.
7-Dec-03		
	Include a "Forget your password" option at log-in	
	Can a button be inserted to Copy the results table onto the clipboard?	Joel C.G.
	cell labels in output should be aligned to bottom left	Joel C.G.
	I want to be able to use the data in SPSS Analysis package. To make this possible, there needs to be numbers assigned to the radio button options, and the number should be put below the radio button question.	Joel C.G.
	Create an Executive Summary page of results for a Feedback Request	Nuttall
	The UK is very sensitive about privacy information, and no private information could be stored on a system which was kept at MIT. It would be necessary to find a way to prevent the privacy data from being stored here.	
	Make a "Text Box Editor" which opens a window that allows easy editing of text for text boxes.	Joel C.G.
	Create LOCK feature, so that objects can't be changed once they have been used in a survey	
	Change navigation to allow easier Edit, create from, etc	Wouter
	There needs to be a way to decrease the number of objects listed (surveys, classes, etc).. Is it possible to just show those that are relevant to the user?	Betty
4-Jan-04		
	There needs to be general email functionality which can be used to communicate to the class	
	There needs to be a Subject Workspace which can be used to edit and create subjects.	
11-Jan-04		
	There should be a Configuration Page for Email	
	make an Error readout page for emails, with status update and log of every person sent to, and the result	
	the "From" field for emails should be changeable so people can respond to the instructor	Brown
	It should be possible to allow people to submit multiple surveys, or keep them to one submission. Implement "Multiple Submissions" mode	Brown

	There is a need for user identification to have the option of being separate from response collection. Create some sort of "Non-associated response" to implement this.	Brown
	There should be an option by which the respondent can be empowered to choose whether or not their name should be attached to the survey	
	There should be the option for students to be given the option to register themselves for a survey (and take a class).	Brown
	The instructor should be able to enable the respondents to edit surveys which they have already submitted.	Brown
	It is useful to have certain data stored automatically for a survey, including the name of the creator, the date on which it was created, when it was last modified, etc.	Brown
	Allow Responses to be deleted, in for the sake of testing or spam.	Brown
	Instructors should be able to determine a period of time within which the survey is active (if that is appropriate)	Brown
	If an option is made to have an answer option of type "Integer", than statistical analysis can be performed on the results	Brown
	In some situations, there might be a fear of instructors tampering with the setting after a response has been collected, and therefore it might be necessary to have the system lock the use of a survey after the first response has come in.	Brown
	It would be very useful to be able to export the data results to a CSV file to be imported to a spreadsheet, etc.	Brown
	Various useful results averages might include avg, std dev, min, max	Brown
	it would be helpful to have a list of those who responded, and a list of those who did not.	Brown
	All authoring tools must be cross-browser compatible	Brown
18-Jan-04		
	allow authentication without identification	Barrett
	there should be a usage note on formatting of dates	Dane
	Is it possible to enforce a minimum threshold for option answers?	Barrett
25-Jan-04		
	make "Response View" pages secure, so that people can't view responses for courses they don't have viewing rights to.	

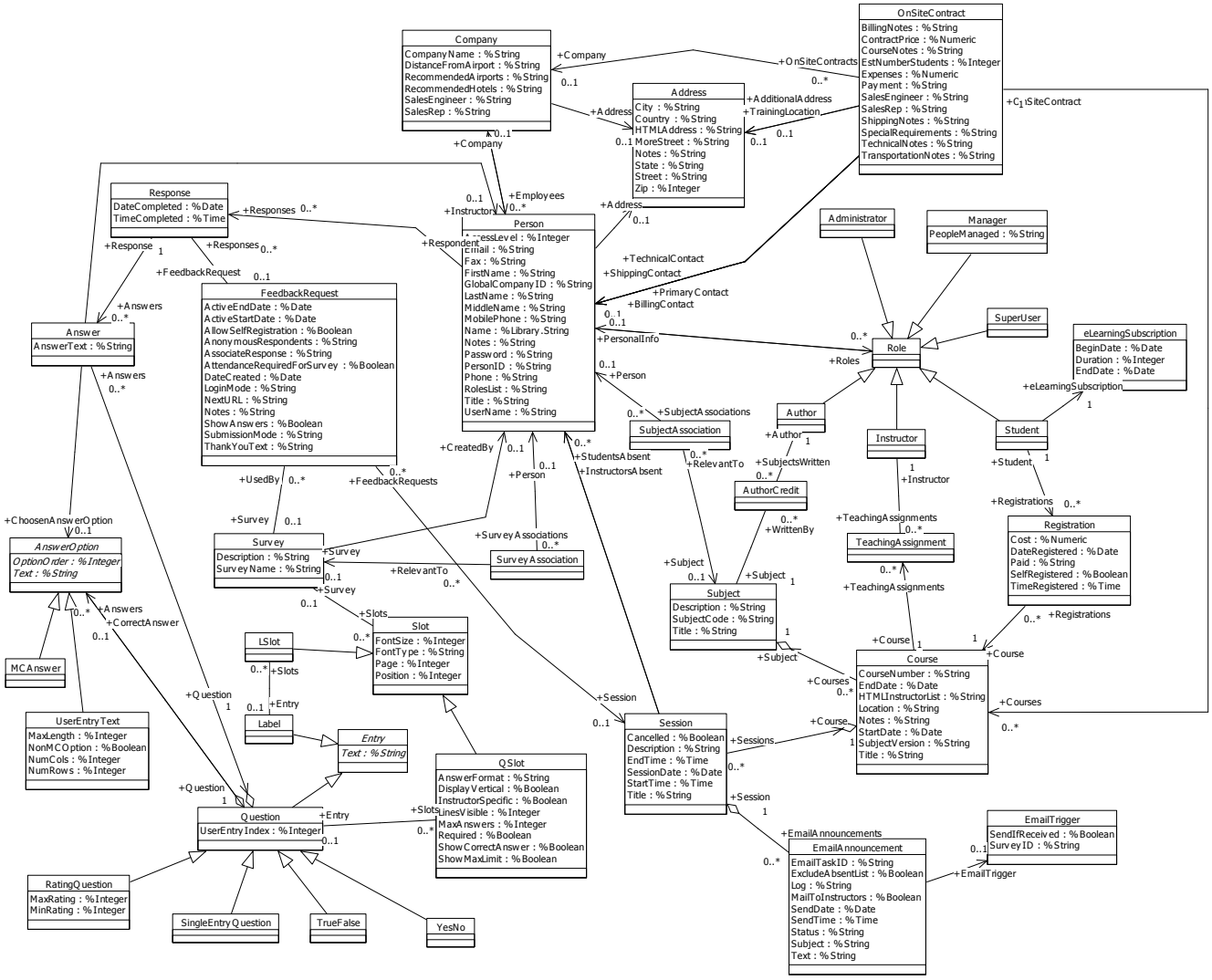
	implement "Back to Session" link on Response set pages	
	separate self-registration responses from pre-registered responses on ResponseList	
	Complete a list of non-respondents in the result set	
	Indicate Self-Registered Students on Course and Session List	
	for deletion confirmation (Subject, Course, Session, Person, etc) display a count of the number of things which will be deleted with the object being deleted	
1-Feb-04		
	Implement Instructor Specific Feedback	
	Make the current nav button a different color in each of the nav bars to improve navigation	
	Implement "Edit Survey" link in Session Workspace for quicker navigation	
	List surveys as "Addable" options in SurveyWorkSpace - include slot preview and usage lists	
	Implement "Active Period", so that the survey is only live between certain dates if the instructor so chooses.	
	put in bars to show percent response on executive summary, to improve visual clarity	
	Change user Password structure and interfaces to improve security	
	implement "Associations" list in Survey Workspace to make it easier for authors to control who sees their surveys	
	automatically associate a new person when they make a survey or subject, to remove the administrators need to do it	
	Test the application on Netscape and make sure that it is entirely cross-platform compatible.	
	Put usage note concerning testing only on IE, so that users know that to date it has only been tested on one platform	
8-Feb-04		
	Create a usage note on frequent saving (and how saving differs from surveys to everything else)	Joel C.G.
	Move save and scrap buttons to upper left-hand corner, as that is the intuitive place for users to look for them	Joel C.G.
	Usage note about start and end dates being option for surveys	Joel C.G.
	enlarge size of logout button to make it more visible	Joel C.G.

	It needs to be possible for respondents to use the systems without any cookies being placed on their machine.	
	Allow Persons to be deleted from the database	
	Create a new help system which is more informative for the user.	
15-Feb-04		
	Is it possible to create rights for a TA, so they can input a class list?	Kirchain
	Confirm before leaving SurveyWorskspace in Netscape, so that they know they need to save their work.	
	Create page showing list of everyone Survey is relevant to	
	Email confirmations for submitted surveys..	Satwik
22-Feb-04		
	Implement "View Person Data" pop-up page to allow easier navigation between objects	
	Create warning / lock for questions used in other surveys, as it is too easy to make a change with ripple effects without knowing it	Joel C.G.
	The response pages need to be easily printable, so that they can be taken and analyzed without being online	
	Is it possible to add special features based on the location of the survey system? There are special features which apply to the corporate but not the academic setting	
	Change code so that Java Applets are not used in any respondent pages - as sending the survey to a large number of people increases the likelihood of someone not having the Java Virtual Machine properly installed	
	Insert a "Clear ALL" button for Instructor/Students in the workspace, to make it easy to clear old or test courses.	
	Add a "Remove All" button for Feedback Requests to make it easier to clear test sessions	
	Add a "Delete All" Button for Reponses to FRs in case it is going to be deleted from the system	
	when using "Create from" with questions and labels, clone the Use objects as a starting point, since there is a good chance that the usage data will be at least partially applicable	
	Create admin page showing people logged into the system to make it easier to administer updates	Hilson
7-Mar-04		
	Create an option in "Create from" for Surveys which allows question reuse or creating new questions	
	Decide how to handle company info, as it differs between the academic and corporate use cases	

	It would be very helpful if this could be used in classes as a primary communication conduit from students to professor, so that the professor doesn't have to wade through emails to find student questions	Newman
	Students might like this as a means of anonymously submitting ad hoc feedback to a professor about things they don't like about a class - a sort of anonymous portal (1 or maybe two questions).	Kim
14-Mar-04		
	Create "New" links for People and Companies (with a way to return to the previous page)	
	Create a pop-up to add payment information when someone is added to a course, to better mirror the corporate training use case	
	Allow Company associations to be removed	
	Allow Contracts, Contacts and Companies to be Deleted	
	This would be a very useful way to replace the paper-based evaluations which are used for every course	Nuttall
	Possible incentive scenarios include having grades (or papers) withheld until the survey is complete, or perhaps the entire class doesn't get their grades until 2/3 of the class fills out the survey	Nuttall
	At the Judge, each student has an ID, which the professor is not allowed to know (the correlation between number and student). Perhaps there is away to use the IDs to keep anonymous responses from the professors point of view.	Nuttall
	this could be a great tool for informal feedback.	Nuttall
	At the university of Cambridge, the drivers of the feedback process are administrative, not academic. The professors are stakeholders and recipients of feedback, but not the drivers. It will be important the tool serve the needs of the administrators, since they would have to give buy-in for it to go into use.	Nuttall
	It is important to have a strategic plan for long term storage if the collection will be through electronic means, since systems change over time.	Ash
	To limit misuse of the system, it might be important to include the assumptions in the reporting section, and make it to that the assumptions and results cannot be separated.	Ash
	In order for this, or any system to take effect, it must have excellent documentation. Documentation is the place where most new systems fall flat on their face, and therefore fail to gain a user following.	Ash

	There needs to be creative ways to structure incentives, since participation grades are normally not given in the UK	Cruickshank
	Feedback is normally collected at the end of each term	Sparling
	The same form is always used, so it makes sense to have something that can be used over and over	Sparling
	There are many instructors listed on one form, and the data pertaining a certain instructor should only go to them (normally it is hand-coded).	Sparling
	It needs to be possible to have the feedback forms be totally anonymous	Sparling
	There is a challenge of being able to broadcast all of the results to all professors involved in a course, without sending out the negative responses about some professors so that others can read and see them.	Sparling
	It should be possible to have simple question formatting, with an advanced option for those people who want the additional features	Williams
21-Mar-04		
	Include a link to OnSiteContract from CourseWorkspace to improve work flow	
	Include a link to courseworkspace from sessionworkspace because it is more intuitive for organization	
	Autopopulate Course Title with Subject Title, as it is often similar (if not the same)	Solon
	Add "Client" calculated field based on Contract for Course Listing	Solon
	Calculated "Location Name" field in OnSiteContract class (and CSP page), having <Company> <City> <State>	
	Usage note on auto saving (workspaces), as it is not intuitive in a web application	Solon
	Allow "Please explain" to be on a separate line in browser rendering	
	Create a "Create Session" link in Course Workspace, to enable workflow	
	Create a "Create Course" link in Subject Workspace, to enable workflow ease	
	Create Import / Export of Surveys using XML for portability across feedback system installations	
	Put Sessions inside Course Workspace, since that is the logical container for the Session objects	

Appendix B – Complete System UML Diagram



Appendix C – Software Classes

Feedback.Address

serial class **Feedback.Address** extends %SerialObject
Address serial class, used to store the address of a person.

Properties

- property **City** As %String
  City of Address
- property **Country** As %String
  Country of Address
- property **HTMLAddress** As %String [Calculated;]
  function which displays the address in HTML format
- property **MoreStreet** As %String
  street of Address
- property **Notes** As %String (MAXLEN = 1500)
  Additonal information pertaining to this address
- property **State** As %String
  State of Address
- property **Street** As %String
  street of Address
- property **Zip** As %Integer
  Zip of Address

Methods

- method **HTMLAddressGet()** returns %String
  method to override the Get method for HTMLAddress since it is a calculated method

---

Feedback.Administrator

persistent class **Feedback.Administrator** extends Feedback.Role
SQL Table Name:
The Administrator role is one that can creaste surveys, add students and instructors to a course, or do other administrative tasks.

Feedback.AlphabeticalList

persistent class **Feedback.AlphabeticalList** extends %Persistent
SQL Table Name:

This is a list of objects in which the inserted objects are all placed alphabetically.

Properties

• list property **List** As %String
    This is the actual list of objects

Methods

• method **InsertIntoList(NewItem)** returns %Status
    This adds an item into the list, in alphabetical order

Feedback.Answer

persistent class **Feedback.Answer** extends %Persistent
SQL Table Name:
Child of Question. The Answer object represents the answers given to a single discrete question. This should be a single AnswerIndex showing the M.C. option selected, and text (if a UserEntry object was used). NOTE: if the question is a multi-select question, then there should be multiple Answer objects with the same QuestionID, where each object has a different AnswerIndex

Properties

• property **AnswerText** As %String (MAXLEN = 1000)
    The text typed by a user in the UserEntry field of a question
• property **ChoosenAnswerOption** As Feedback.AnswerOption
    This is the AnwerOption choosen by the student. If the answer choosen was a UserEntry box, then there will be text in the AnswerText property of this object
• property **Instructor** As Feedback.Person
    this properly is used (and indexed) to indicate that this was an instructor specific survey, and to the instructor specified by the student was the instructor with this ID. This will only be filled in when the Survey has the property "InstructorSpecific" set to true
• relationship **Question** As Feedback.Question [Inverse = Answers; Cardinality = parent;]
    Question (Parent relationship) to which this object is the an Answer
• relationship **Response** As Feedback.Response [Inverse = Answers; Cardinality = one;]
    One relationship - the Response in which the Answer was given

Queries

• query **AnsOptCountsByQandFR(FeedbackRequestID As %String, QuestionID As %String)**
returns the count of each AnswerOption related to a given question and FeedbackRequest
• query **AnsOptCountsByQandFRandInstr(FeedbackRequestID As %String, QuestionID As %String, InstrID As %String)**
returns the count of each AnswerOption related to a given question and FeedbackRequest
• query **NumAnsByQandFRandInstr(FeedbackRequestID As %String, QuestionID As %String, InstrID As %String)**
returns count of the number of distinct Instructor specific responses for a given question in a given FeedbackRequest
• query **NumAnsOptByQandFRandInstr(FeedbackRequestID As %String, QuestionID As %String, AnswerOptionID As %String, InstrID As %String)**
returns count of the number of times AnswerOption was choosen for in Instructor in a given question in a given FeedbackRequest
• query **NumAnswersByQuestionAndFR(FeedbackRequestID As %String, QuestionID As %String)**
returns count of the number of distinct responses for a given question in a given FeedbackRequest
• query **NumSpecificAnsOptByQAndFR(FeedbackRequestID As %String, QuestionID As %String, AnswerOptionID As %String)**
returns count of the number of times AnswerOption was choosen for a given question in a given FeedbackRequest
• query **UserTextByQAndFR(FeedbackRequestID As %String, QuestionID As %String, AnswerOptionID As %String)**
returns text of all AnswerTexts given for this question in a given FeedbackRequest
• query **UserTextByQandFRandInstr(FeedbackRequestID As %String, QuestionID As %String, AnswerOptionID As %String, InstrID As %String)**
returns text of all AnswerTexts given for this question in a given FeedbackRequest

Indices

• index **InstructorIndex** on Instructor
• index **ResponseIndex** on Response

Feedback.AnswerOption

abstract persistent class **Feedback.AnswerOption** extends %Persistent
SQL Table Name:
Abstract class which is the model for both kinds of Answer Options - UserEntry or Multiple-Choice

Properties

• property **OptionOrder** As %Integer
    this parameter tells the order in which this AnswerOption object is listed in the question. This is for the sake of giving the survey creator full flexibility in how the survey is displayed
• relationship **Question** As Feedback.Question [Inverse = AnswerOptions; Cardinality = parent;]
    Parent of AnswerOption - the question to which this is a possible answer
• property **Text** As %String (MAXLEN = 500)
    The Text of the answer (Mulitiple-Choice or UserEntry), which is in effect what the student will be "choosing"

Methods

• method **InsertAfter(optord As %Integer)** returns %Status
    This method sets the order of the current AnswerOption object. It takes 1 parameter (optord), sets ..OptionOrder to optord+1, and adds 1 to the OptionOrder values of all other AnswerOptions related to this Question NOTE: ..Question must be set for this to work - if it is not, then this method will return a zero. If this is to be inserted into the first place in the question, then pass 0 as the parameter
• method **MoveDown()** returns %Status
    This method moves the AnswerOption down one position. NOTE: ..Question must be set for this to work - if it is not, then this method will return a zero. If this is already the first AnswerOption, then there will be no change
• method **MoveUp()** returns %Status
    This method moves the AnswerOption up one position. NOTE: ..Question must be set for this to work - if it is not, then this method will return a zero. If this is already the first AnswerOption, then there will be no change
• method **SetNextPosition()** returns %Status
    This method looks at all AnswerOption objects related to this ..Question, and finds the maximum Order value for all objects. Then it sets ..OptionOrder to the greatest location value + 1. If the code runs without issues, a 1 is returned, otherwise a 0 is returned Note: ..Question must be set prior to calling this method

---

Feedback.Author

persistent class **Feedback.Author** extends Feedback.Role

SQL Table Name:
This is the role of a person who writes the material used in a Subject Learning Object

Properties

• relationship **SubjectsWritten** As Feedback.AuthorCredit [Inverse = Author; Cardinality = many;]
one to Many relationship which lists the objects identifying the Subjects to which the Author has contributed

---

Feedback.AuthorCredit

persistent class **Feedback.AuthorCredit** extends %Persistent
SQL Table Name:
This is a persistant class created to allow a many to many relationship between authors and subjects. Specifically, 1 Author can write (and receive credit for) many Subjects, and 1 Subject can be written by (and thus credit is due to) several Authors.

Properties

• relationship **Author** As Feedback.Author [Inverse = SubjectsWritten; Cardinality = one;]
The Author to whom Credit is due for work contributed towards this subject
• relationship **Subject** As Feedback.Subject [Inverse = AuthorCredits; Cardinality = one;]
The subject connected to the Author who is due credit for helping to author it

Indices

• index **AuthorIndex** on Author
• index **SubjectIndex** on Subject

Feedback.Company

persistent class **Feedback.Company** extends %Persistent
SQL Table Name:
This class is to keep track of company information (for corporate training applications)

Properties

• property **Address** As Feedback.Address
the address of the company
• property **CompanyName** As %String [Required;]
the name of the company
• property **DistanceFromAirport** As %String
the distance that the company site is from the airport
• relationship **Employees** As Feedback.Person [Inverse = Company; Cardinality = many;]
this is a one to many relationship with person objects, where the people work for this company
• relationship **OnSiteContracts** As Feedback.OnSiteContract [Inverse = Company; Cardinality = many;]
this is a one to many relationship between the company, and the contracts created by that company for OnSiteTraining
• property **RecommendedAirports** As %String (MAXLEN = 500)
Airports which should be used when visiting this customer
• property **RecommendedHotels** As %String (MAXLEN = 500)
Hotels which this company recommends be used when ISC people visit
• property **SalesEngineer** As %String
The Sales Engineer currently working with the company
• property **SalesRep** As %String
the sales rep currently working with the company

Feedback.Course

persistent class **Feedback.Course** extends %Persistent
SQL Table Name:
A Course is a time-bound instance of a Subject, i.e. it is descrete period over which the Subject is taught. Examples might be: Subject = Physics 101 Course = Fall 2003 or Subject = U.S.History 202 Course = Summer 2001 Therefore, is a set of materials (Subject) is taught several times, then each time it is taught, that represents a seperate Course (which is a child of Subject).

Properties

• property **CourseNumber** As %String [Calculated;]
The AutoGenerated Course Number (calculated from SubjectCode and %Id())
• property **EndDate** As %Date (FORMAT = 5)
when the Course ends
• property **HTMLInstructorList** As %String [Calculated;]
function which displays the Instructor List with HTML line breaks
• property **Location** As %String
Location of the training
• property **Notes** As %String (MAXLEN = 500)

Notes associated with this class

• relationship **OnSiteContract** As Feedback.OnSiteContract [Inverse = Courses; Cardinality = one;]

This is used for Corporate Training Applications, in which there is a contract used for training Courses

• relationship **Registrations** As Feedback.Registration [Inverse = Course; Cardinality = many;]

One to Many relationship in which the Registration points to a single student who is registered for this Course

• relationship **Sessions** As Feedback.Session [Inverse = Course; Cardinality = children;]

Children of Course - the Sessions break up the Course into smaller pieces based on Time, or Material or both (It makes no sense for Sessions to exist without the context of a Course)

• property **StartDate** As %Date (FORMAT = 5)

When the Course begins

• relationship **Subject** As Feedback.Subject [Inverse = Courses; Cardinality = parent;]

Parent of Course - the subject is the highest level of Learning Material classification

• property **SubjectVersion** As %String

Version of the Subject (major and/or minor revision) used for this Course

• relationship **TeachingAssignments** As Feedback.TeachingAssignment [Inverse = Course; Cardinality = many;]

The one to Many relationship which points to the objects joining an Instructor to this Course

• property **Title** As %String (MAXLEN = 250)

Course Title - Could be the same as the Subject Title, or it could reflect the period in which the Course is given

Methods

• classmethod **CalcHTMLInstructorList(id As %Library.Integer)** returns %Library.String

name calculation

• method **Cancel()** returns %Status

Cancels the course, and consequently cancels each session by callings its Cancel() method (with EmailInform set to false in the session methods). It also sends out an email to all students and instructors associated with the course if the Cancel() method was called with "EmailInform=True", informing them that the course was canceled (the text is given in the Body argument, otherwise a default message is sent)

• classmethod **CourseNumberCalc(ID As %String, Code As %String)** returns %String
    this is a class method used for the Calculated property CourseNumber
• method **CourseNumberGet()** returns %String
    method to override the Get method for CourseNumber since it is a calculated method
• method **HTMLInstructorListGet()** returns %String
    method to override the Get method for HTMLInstructorList since it is a calculated method

Queries

• query **CourseList()**
    This is a list of all the Courses in the System
• query **CourseListByPerson(PersonID)**
    This is a list of all the Courses in the System based on relation to PersonID through Subject

Indices

• index **OnSiteContractIndex** on OnSiteContract
• index **SubjectIndex** on Subject

Feedback.CSPSuperPage

class **Feedback.CSPSuperPage** extends Feedback.CspUtils
This is a page that contains the OnPreHTTP method that is inherited by every CSP page.

Class Parameters

• parameter **AccessLevel** [= 0]

Methods

• classmethod **CheckAccess(PageAccessLevel)** returns %Boolean
    this checks the access level on the page, and determines if the user is logged in an should be able to access this page Access Stucture: 0 - Open access 1 - Student 2 - Instructor and Author 4 - Manager 6 - Administrator 8 - SuperUser
• final classmethod **OnPreHTTP()** returns %Boolean

Event handler for **PreHTTP** event: this is invoked before the HTTP headers for a CSP page have been sent. All changes to the %CSP.Response class, such as adding cookies, HTTP headers, setting the content type etc. must be made from within the OnPreHTTP() method. Also changes to the state of the CSP application such as changing %session.EndSession or %session.AppTimeout must be made within the OnPreHTTP() method. It is prefered that changes to %session.Preserve are also made in the OnPreHTTP() method as this is more efficient, although it is supported in any section of the page. Return **0** to prevent **OnPage** from being called.

Feedback.CspUtils

class **Feedback.CspUtils** extends %CSP.Page
this is a utility wrapper class, used to hold methods which wil create HTML to fulfill various needs on CSP pages

Methods

• classmethod **CSAddCourseToContract(CourseID, ContractID)** returns nothing.
    this method takes a CourseID, and assigns that course to a Contract
• classmethod **CSAddOption()** returns nothing.
    This is the method that Adds a new AnswerOption to a a question
• classmethod **CSAddRole(PersonID, role)** returns nothing.
    this is a method which adds a role to a person object
• classmethod **CSAddSlot(EntryID, type, create)** returns nothing.
    This is the method that adds a slot into a survey EntryID - is "" if an entry is to be made from scratch, otherwise it has the EntryID of the Entry in the database to use type - either 'Label' or 'Question'
    create - this is set when a copy of the object with EntryID is to be made, rather than using the object which already exists
• classmethod **CSAddUserEntry()** returns nothing.
    This is the method that Adds a User Entry to the question
• classmethod **CSAssignBillingToPerson(PersonID, BillingID)** returns nothing.
    this method takes a person, assigns a billing contact to them
• classmethod **CSAssignCompanyToContract(ContractID, CompanyID, CopyAddress)** returns nothing.
    this method takes a Contract, assigns a company to it, and updates the Contract's Location address with the company address
• classmethod **CSAssignCompanyToPerson(PersonID, CompanyID, CopyAddress)** returns nothing.
    this method takes a person, assigns a company to them, and updates the person's address with the company address

• classmethod **CSAssignContactToContract(ContractID, ContactID, ContactType)** returns nothing.
    this method takes a person, assigns a billing contact to them
• classmethod **CSChangeAbsentStatus(PersonId, role, NewStatus, SessionId)** returns nothing.
    this is a method that moves a student or instructor between absent and attending session lists
• classmethod **CSChangeCourseTitle(CourseID, Title)** returns nothing.
    this method changes the title of the course which is currently being worked on in CourseWorkspace
• classmethod **CSChangeSessionTitle(SessionID, Title)** returns nothing.
    this method changes the title of the session which is currently being worked on in SessionWorkspace
• classmethod **CSChangeSubjectAssociations(BoxName, Checked, PersonID)** returns nothing.
    this method either creates or destroys SubjectAssociation objects
• classmethod **CSChangeSubjectTitle(SubjectID, Title)** returns nothing.
    this method changes the title of the subject which is currently being worked on in SubjectWorkspace
• classmethod **CSChangeSurveyAssociations(BoxName, Checked, PersonID)** returns nothing.
    this method either creates or destroys SubjectAssociation objects
• classmethod **CSChangeSurveyName(Name, Description)** returns nothing.
    this method changes the name of the survey which is currently being worked on in SurveyWorkspace
• classmethod **CSChangeToAnonymous(PerID="")** returns nothing.
    this method changes the user's login to Annonymous
• classmethod **CSCheckUserName(UserName)** returns nothing.
    this method searches the database for the existance of a the username. If it exists, then a 1 is returned, if it does not, a -1 is returned
• classmethod **CSDeleteCourse(CourseID)** returns nothing.
    This is the method that deletes a Course from the Database
• classmethod **CSDeleteEmail(EmailID)** returns nothing.
    This is the method that deletes an Email from the database
• classmethod **CSDeleteFeedbackRequest(FeedbackRequestID)** returns nothing.
    This is the method that deletes a Feedback Request from a session
• classmethod **CSDeleteOption(index)** returns nothing.
    This is the method that deletes an option from a question
• classmethod **CSDeletePerson(PersonID)** returns nothing.
    This is the method that deletes a Person from the Database
• classmethod **CSDeleteResponse(ResponseID)** returns nothing.
    This is the method that deletes a Response from a session
• classmethod **CSDeleteSession(SessionID)** returns nothing.
    This is the method that deletes a Session from the Database

• classmethod **CSDeleteSlot(index)** returns nothing.
    This is the method that deletes a slot from a survey
• classmethod **CSDeleteSubject(SubjectID)** returns nothing.
    This is the method that deletes a Session from the Database
• classmethod **CSDeleteSurvey(SurveyID)** returns nothing.
    This is the method that deletes a slot from a survey
• classmethod **CSHTMLWordWrap(Text, Length)** returns %String
    this is a method which adds a role to a person object
• classmethod **CSMoveOption(index, direction)** returns nothing.
    This is the method that moves the order of the options in a question
• classmethod **CSMoveSlot(index, direction)** returns nothing.
    This is the method that moves the order of the slots on a survey
• classmethod **CSRegisterInstructor(CourseID, PersonID)** returns nothing.
    this method registers a Instructor in a Course
• classmethod **CSRegisterStudent(CourseID, PersonID)** returns nothing.
    this method registers a Student in a Course
• classmethod **CSRemoveCourseFromContract(ContractID, index)** returns nothing.
    this method takes a ContractID, and removes the course at "index" from Courses
• classmethod **CSRemoveUserEntry()** returns nothing.
    This is the method that Adds a User Entry to the question
• classmethod **CSSaveSurvey(PersonID)** returns nothing.
    this method saves the survey which is currently being worked on in SurveyWorkspace
• classmethod **CSSelfRegister(PersonID, CourseID)** returns nothing.
    this is a method which is called when the respondent chooses to self-register for a course in order to take a survey
• classmethod **CSSendEmail(EmailID)** returns nothing.
    This is the method that Sends an Email immediately
• classmethod **CSUnregister(role, RegisterID)** returns nothing.
    this is a method removes an Instructor or Student from a Course
• classmethod **CSUpdateSubjectAssociations(PersonID)** returns nothing.
    this method updates the subjectAssociations page by writing JavaScript code to update the objects on the page form
• classmethod **CSUpdateSurveyAssociations(PersonID)** returns nothing.
    this method updates the surveyAssociations page by writing JavaScript code to update the objects on the page form
• classmethod **CSUpdateText(index, text)** returns nothing.
    This is the method that updates Slot of Option Text following an onBlur event from a text box
• classmethod **CSUseSurvey(SurveyID, SessionID)** returns nothing.
    This is a method that adds a Survey use to a Session
• classmethod **GetParmValue(ParmName)** returns %String
    this is a method which adds a role to a person object
• classmethod **HTMLStyle()** returns nothing.

This is a kludge to try to make unified style definitions until I learn to use style sheets

- classmethod **JSAlertVerificationLogin()** returns nothing.
  This is a method which writes the JavaScript that creates the alert explaining a Verification login
- classmethod **JSCheckFormInputObjects()** returns nothing.
  this contains javascript methods necessary to see if form input objects have been selected or not
- classmethod **JSExitOnEscape()** returns nothing.
  This is a method which writes the JavaScript that opens a window to show a Slot use.
- classmethod **JSNavigateResponses()** returns nothing.
  this method implements the JavsScript code to jump between the response pages
- classmethod **JSPersonPopup()** returns nothing.
  This is a method which creates the pop-up with a person's info
- classmethod **JSPreviewSurvey()** returns nothing.
  This is a method which writes the JavaScript that opens a preview window showing the survey.
- classmethod **JSToggleDIV()** returns nothing.
  This is a method which writes the JavaScript needed to toggle DIV blocks on and off
- classmethod **JSToggleResponse()** returns nothing.
  this is a method which needs to be called from any CSP page that renders question objects which can be toggled. This will be toggle questions and labels
- classmethod **JSVerifyListSelection()** returns nothing.
  this is a method to write the javascript method used to verify that an element on a list has been selected prior to trying to edit it or build from it
- classmethod **JSVerifyResponse()** returns nothing.
  this is a method which needs to be called from any CSP page that renders question objects. It contains the code to validate the responses
- classmethod **JSViewSlot()** returns nothing.
  This is a method which writes the JavaScript that opens a window to show a Slot use.
- classmethod **JSVisibility()** returns nothing.
  This is a method which writes the JavaScript for the functions that change the visibility of objects

Queries

- query **RegisteredForCourse(FeedbackRequestID As %String, PersonID As %String)**

this is a query which accepts a FeedbackRequest object, and a PersonID, and it works through the chained relationship to find if that person is registered for the Course in which that FeedbackRequest is used

---

Feedback.eLearningSubscription

serial class **Feedback.eLearningSubscription** extends %SerialObject
Serial Class - an InterSystems eLearning Subscription, this object is used for the sake of customer tracking, etc.

Properties

• property **BeginDate** As %Date (FORMAT = 5)
Date that subscription begins
• property **Duration** As %Integer
duration for which this subsciption lasts (in days)
• property **EndDate** As %Date [Calculated;]
date that subscription ends (calculated from Duration and BeginDate)

Methods

• classmethod **EndDateCalc(Duration As %Integer, StartDate As %Date)** returns %Date
this is a class method used for the Calculated property EndDate
• method **EndDateGet()** returns %Date
method to override the Get method for EndDate since it is a calculated method

---

Feedback.EmailAnnouncement

persistent class **Feedback.EmailAnnouncement** extends %Persistent
SQL Table Name:
Child of Session. The purpose of the EmailAnnouncement class is to create specifc instances of communication points which the instructor wishes to have with the class. These can all be specified at once, and then sent out using the Cache scheduler.

Properties

• property **EmailTaskID** As %String
this is a pointer to the %System.Task object which is scheduled to send the email specified by this object BEN: change this to a pointer that can be used

• property **EmailTrigger** As EmailTrigger
    Serial Class which contains the details on how the email is to be triggered (the absence of this class's inclusion means that the email is sent to all students (and possibly Instructors).
• property **ExcludeAbsentList** As %Boolean [InitialExpression = "0";]
    When set to FALSE, the email is sent to all students registered for the class (and instructors too if the "MailToInstructors" flag is set). If TRUE, then the students on the StudentsAbsent list are exluded from the mailing.
• property **Log** As %String (MAXLEN = 5000)
    This is a log string, stored in the form of HTML Rows, such that it can be displayed easily. The content is the first and last names, email and returned status of messages sent
• property **MailToInstructors** As %Boolean
    Boolean field indicating whther or not the instructors in the InstructorList should be included in the mailing
• property **SendDate** As %Date (FORMAT = 5) [Required;]
    Date email is to be sent
• property **SendTime** As %Time
    Time email is to be sent
• relationship **Session** As Feedback.Session [Inverse = EmailAnnouncements; Cardinality = parent;]
    Parent of EmailAnnouncement - Session with which this announcement is associated
• property **Status** As %String (VALUELIST = ,Queued,Sent,Error) [InitialExpression = "Queued";]
    This is a status indicator showing where or not the email has been sent. The three options are: "Queued", which means a task has been created and the task has not fired yet; "Sent" which means that the email was sent without any errors, and "Error" which means that sending email was attempted, but an error occurred
• property **Subject** As %String (MAXLEN = 200) [Required;]
    Text for the Subject Line of the Email
• property **Text** As %String (MAXLEN = 1000) [Required;]
    Text for the body of the Email

Methods

• classmethod **DeleteFinishedTasks()** returns %Status
    This is a class method which is run once a day for the purpose of cleaning up any tasks that have already sent out their emails. Once an email has been sent, its associated task is deleted.
• classmethod **Initialize()** returns %Status

This only needs to tun once an a machine. This class method creates a task that runs every day and looks for exired email tasks to they can be deleted.

- classmethod **SendAnnouncement(emailID As %String)** returns %Status
  this is a classmethod which actually sends the emails to the recipients. This classmethod is called by the scheduled email task. This method opens the appropriate EmailAnnoucnement object, and extracts the information necessary to send the emails to the session attendees
- method **SendNow()** returns %Status
  method to cause an email to go out now to the proper recipiants

Queries

- query **FindQueuedEmails(SessionID As %String)**
  this query is used to find all emails associated with a certain session which have the status of "Queued"
- query **FindSentEmails(SessionID As %String)**
  this query is used to find all emails associated with a certain session which do not have the status of "Queued"

Feedback.EmailTrigger

serial class **Feedback.EmailTrigger** extends %SerialObject
serial class - used to pinpoint a specific Survey and see if it has been received by a specific user so see whether or not to trigger a future email.

Properties

- property **SendIfReceived** As %Boolean
  boolean - send the email id the SurveyID survey has been received from a given student in the class list.
- property **SurveyID** As %String
  survey used to trigger sending an email to a particular student on the class list.

Feedback.Entry

abstract persistent class **Feedback.Entry** extends %Library.Persistent
SQL Table Name:
Abstract class from which Label and Question inherit. An Entry is a discrete entity on a Survey, and it always contains Text (whether that be Label text, or a Question text) It is in a one to many relastionship with Slot

Properties

• property **Text** As %String (MAXLEN = 1000) [Required;]
Text of the entry (required) - which is a Heading or explaination if the Entry is a Label object, or is the Question statement if the Entry is a Question object

---

Feedback.FeedbackRequest

persistent class **Feedback.FeedbackRequest** extends %Persistent
SQL Table Name:
The FeedbackRequest object links the Many-to-Many relationship between Sessions and Surveys. Since a Survey does not have to be tied to a specific Session, there needs to be a way to link the two, which is accomplished through the FeedbackRequest

Properties

• property **ActiveEndDate** As %Date (FORMAT = 5)
This is a date variable which shows the last day that this feedback request can be filled (ie a servey be answered)
• property **ActiveStartDate** As %Date (FORMAT = 5)
This is a date variable which shows the first day that this feedback request can be filled (ie a servey be answered)
• property **AllowSelfRegistration** As %Boolean
this is a boolean flag which records whether or not a person is allowed to self-register. When they self-register, they are automatically registered for the course, and therefore allowed to take the survey
• array property **AnonymousRespondents** As %String
This is an array of all of the people who have logged in to take this survey, but had their responses associated with the "Anonymous" profile. This is only an option when Associate Response is not set to "Yes". In this array, the Key is the %Id of the Person Object who logged in, and the Element is the number of submissions which they have given
• property **AssociateResponse** As %String (VALUELIST = ,Yes,No,StudentChoice) [InitialExpression = "Yes"; Required;]
This is a property which dictates whether or not logged in users are associated in the database with the answers which they provide. The three options are "Yes" (associate the users with the response they give), "No" (maintain a seperate list of users who submit responses, but do not associate their answers with those responses) and "StudentChoice" (the option is given to the student as to whether or not they want their response to be associated with their identity
• property **AttendanceRequiredForSurvey** As %Boolean

This is a boolean field, which is used to indicate that students registered for a course are not allowed to take surveys if they have been marked absent from the session to which the survey has been attached. This is intended for use with class participation grading based on survey responses

• property **DateCreated** As %Date (FORMAT = 5)

This is when this object was created

• property **LoginMode** As %String (VALUELIST = ,LoginRequired,AnonymousOption,AlwaysAnonymous) [InitialExpression = "LoginRequired"; Required;]

This field controls the log-in requirements for a particular FeedbackRequest. There are three values which are options: "Required" (users must log-in in order to fill in the survey) "AnonymousOption" (users will have a choice of logging in or answering the survey anonymously) and "AlwaysAnonymous" (the FeedbackRequest always logs the user as a Guest, and there is no option for the user to log-in)

• property **NextURL** As %String (MAXLEN = 500)

This field contains the URL to which the student is directed upon completion of the survey. If this field is left blank, then the student will be brought to a default URL which shows a "Thank You" message for filling out the survey (this text can be specialized through the "ThankYouText" property

• property **Notes** As %String (MAXLEN = 500)

This is a text field which can be used to view notes given for a feedback request

• relationship **Responses** As Feedback.Response [Inverse = FeedbackRequest; Cardinality = many;]

• relationship **Session** As Feedback.Session [Inverse = FeedbackRequests; Cardinality = one;]

Session with which the accociated survey is used

• property **ShowAnswers** As %Boolean

Boolean Property. If True, then after a survey Response has been received, the student will be shown a page showing the correcct answers to all of the questions (as well as the answers they filled in). If this field is set to False, then the student will see a thank you page after filling out the survey

• property **SubmissionMode** As %String (VALUELIST = ,Single,Changeable,Multiple) [InitialExpression = "Single"; Required;]

This is a property which signifies the restraints placed on the students concerning multiple submissions. The options are "Single" (only one submission by each student is allowed), "Changeable" (once a survey has been submitted, it may be revisted by the student and changed), and "Multiple" (multiple submissions are allowed)

• relationship **Survey** As Feedback.Survey [Inverse = UsedBy; Cardinality = one;]
Survey to be used with the associated Session
• property **ThankYouText** As %String (MAXLEN = 1000) [InitialExpression = "Thank you very much for taking the time to fill out this survey";]
Optional field which contains the text that will be used to populate the Thank You confirmation page shown to the student following the submission of a survey page

Methods

• method **SurveyURL(ServerName)** returns %String
This is a method that accepts the Server Name, and spits out the full URL needed to access this specific FeedbackRequest via a web browser

Queries

• query **FeedbackRequestList()**
This is a query used to find all survey requests in the database so they can be displayed

Indices

• index **SessionIndex** on Session
• index **SurveyIndex** on Survey

Feedback.Instructor

persistent class **Feedback.Instructor** extends Feedback.Role
SQL Table Name:
an Instructor role is an individual respensible for the disemination of Learning Material to the Students.

Properties

• relationship **TeachingAssignments** As Feedback.TeachingAssignment [Inverse = Instructor; Cardinality = many;]
One to Many relationship pointing to the objects associated with Courses that this instructor is connected to in a teaching capacity

Methods

• method **IsTeacherFor(CourseId)** returns Feedback.TeachingAssignment
This method received one parameter (CourseId), and checks whether or not any of the teachingassignments associated with this instructor are connected to the course with the passed ID. If so, then the TeachingAssignmentId is returned, otherwise "" is returned

---

Feedback.Label

persistent class **Feedback.Label** extends Feedback.Entry
SQL Table Name:
Label class is used for creating a text label of some sort within a survey. This text has an associated font Size, which is the HTML size used to display the text in the survey

Properties

• relationship **Slots** As Feedback.LSlot [Inverse = Entry; Cardinality = many;]
is a Many relationship to an object which associates this Entry with a specific survey in which it is supposed to appear.

Feedback.LSlot

persistent class **Feedback.LSlot** extends Feedback.Slot
SQL Table Name:
This class extends slot with the purpose of adding properties specifically needed for a single instance use of a label object

Properties

• relationship **Entry** As Feedback.Label [Inverse = Slots; Cardinality = one;]
The Survey (One relationship) in which the accociated Entry is placed

Methods

• method **RenderHTML(SessionID As %String = "")** returns nothing.
this method is used to render a label slot stored in the database as HTML
• method **RenderResponseHTML(SessionID, ViewerID)** returns nothing.
this is a dummy method to allow polymorphism with the labels for this method

Queries

• query **SlotList()**
This is a query used to find all Label slots used in the database

Indices

• index **EntryIndex** on Entry

Feedback.Manager

persistent class **Feedback.Manager** extends Feedback.Role
SQL Table Name:
The Manager role defines the properties and methods necessary for a manager. This role is used for supervisors of Instructors to enable them to view the survey given on specific Instructors.

Properties

• list property **PeopleManaged** As %String
this is an Array where the key is the UserName of the Instructors managed by the Manager.

Feedback.MCAnswer

persistent class **Feedback.MCAnswer** extends Feedback.AnswerOption
SQL Table Name:
Extends AnswerOption. A Mulitiple-Choice Answer option. This is a sinlge answer option shown to the user for the associated question. Example: If the question has 4 answers, A, B, C, and D, then there will be 4 MCAnswer objects related to the Question, where the Text of each object is set to one of the 4 options.

Feedback.OnSiteContract

persistent class **Feedback.OnSiteContract** extends %Persistent
SQL Table Name:
This class holds the information pertaining to training contracts which are to be given at the customer site.

Properties

• property **AdditionalAddress** As Feedback.Address

This is to be used for any other Address which should be associated with this Contract (i.e. if there is a special billing address different from the person who is being billed)

- property **BillingContact** As Feedback.Person
  This is a person who should be contacted for the billing
- property **BillingNotes** As %String (MAXLEN = 1500)
  This is where notes relating to the billing can be put
- relationship **Company** As Feedback.Company [Inverse = OnSiteContracts; Cardinality = one;]
- property **ContractPrice** As %Numeric
  This is the amount due for this contract (not including instructor expenses)
- property **CourseNotes** As %String (MAXLEN = 1500)
  this field hols any notes relating to the course requested by the customer
- relationship **Courses** As Feedback.Course [Inverse = OnSiteContract; Cardinality = many;]
  These are the Courses which are put onto the Contract
- property **EstNumberStudents** As %Integer
  this is the initial estimation of the number of students which will be in the class
- property **Expenses** As %Numeric
  the expenses of the instructor during the visit
- property **Payment** As %String
  the payment info goes here
- property **PrimaryContact** As Feedback.Person
  this is the primary contact person for this contract
- property **SalesEngineer** As %String
  this is the active sales engineer who is handling this company at the time of this Contract
- property **SalesRep** As %String
  this is the sales rep servicing this company at the time of the contract
- property **ShippingContact** As Feedback.Person
  This is the information for the person to whom the materials get shipped
- property **ShippingNotes** As %String (MAXLEN = 1500)
  This is where notes relating to the shipping
- property **SpecialRequirements** As %String (MAXLEN = 1500)
  a notes section detailing any special requirements which this customer may have
- property **TechnicalContact** As Feedback.Person
  This is the person who understand the technical aspects of the training to take place
- property **TechnicalNotes** As %String (MAXLEN = 5000)

this is for all questions relating to the technical aspects of teaching the class
• property **TrainingLocation** As Feedback.Address
This is the address where the training is to take place - it can be auto filled from the company information
• property **TransportationNotes** As %String (MAXLEN = 5000)
This holds any notes relating to the transportation details of the trip

Indices

• index **CompanyIndex** on Company

Feedback.Parameters

persistent class **Feedback.Parameters** extends %Persistent
SQL Table Name:

Properties

• property **Name** As %String
• property **Value** As %String

Methods

• classmethod **GetValue(ParmName)** returns %String
this is a class method used for retreiving the value of a parameter object
• classmethod **SetValue(ParmName, ParmValue)** returns %String
this is a class method used for setting the value of a parameter object or creating a new one

Indices

• index **NameIndex** on Name [Data = Value; Unique;]
by storing the Value in the Index, it makes it fast for every look-up

---

Feedback.Person

persistent class **Feedback.Person** extends %Persistent
SQL Table Name:

Class to hold information specific to given individuals within the Learning environment. Due to the challenges of allowing a person to fill more than one role, it is necessary to embed serial objects within person which give the particulars about the role they are filling. A person will be uniquely identified by their UserName.

Properties

- property **AccessLevel** As %Integer [Calculated;]
  This is a calculated property, which determines from this person's relationships what their level of access should be to site content. Student = 1, Instructor & Author = 2, Manager = 4, Administrator = 6
- property **Address** As Address
  Address serial class, used to store the address of a person.
- property **BillingContact** As Feedback.Person
  This is a person object who should be used for the contact for all billing inquiries
- relationship **Company** As Feedback.Company [Inverse = Employees; Cardinality = one;]
  company the person works for (optional)
- property **Email** As %String
  Contact Email of the person
- property **Fax** As %String
  Fax number of Person
- property **FirstName** As %String
  First name of the Person
- property **GlobalCompanyID** As %String
  ISC internal companyID
- property **LastName** As %String
  Last Name of the person
- property **MiddleName** As %String
  middle name of a person
- property **MobilePhone** As %String
  cell or mobile phone of person
- property **Name** As %Library.String [Calculated;]
  Name property
- property **Notes** As %String (MAXLEN = 1500)
  Notes taken on this person
- property **Password** As %String
  password that accompanies UserName
- property **PersonID** As %String [Required;]
  DO NOT SET - INTERNAL USE ONLY: this is a unique property used for IDKey, it is automatically set
- property **Phone** As %String
  primary contact phone of the person

• relationship **Responses** As Feedback.Response [Inverse = Respondent; Cardinality = many;]
• array property **Roles** As Role
    This is an array which hold the "hats" or roles that a person plays. The key to the Array is the name of the role
• property **RolesList** As %String [Calculated;]
    This is a calculated property necessary to be able to easily list the roles that a person has in a CSPBIND Form
• relationship **SubjectAssociations** As Feedback.SubjectAssociation [Inverse = Person; Cardinality = many;]
• relationship **SurveyAssociations** As Feedback.SurveyAssociation [Inverse = Person; Cardinality = many;]
• property **Title** As %String
    The peron's title in their organization
• property **UserName** As %String
    Unique identifier used to identify individual people in the system.

Methods

• method **AddRole(role)** returns nothing.
    This method will add the given role to this person object
• classmethod **CalcAccessLevel(id As %Library.Integer)** returns %Library.String
    name calculation
• classmethod **CalcName(id As %Library.Integer)** returns %Library.String
    name calculation

Queries

• query **CoursesTaken(PersonID As %Library.String)**
    This finds all Course Objects that have a registration with this person ID
• query **EmployeeOf(CompanyID As %Library.String)**
    This finds all Person Objects such that their company has this Company ID
• query **FindAll()**
    This is a query to find all Person objects
• query **FindUserName(UN As %String = "")**
    this query is used to find a user at login time

Indices

• index **FirstNameIndex** on FirstName

• index **IDKEY** on PersonID [IdKey;]
• index **LastNameIndex** on LastName
• index **PersonIDIndex** on PersonID [Unique;]
• index **UserNameIndex** on UserName [Unique;]

Feedback.QSlot

persistent class **Feedback.QSlot** extends Feedback.Slot
SQL Table Name:
This class extends the slot class, with the purpose of adding properties needed to specify characteristics of a single use of a question instance.

Properties

• property **AnswerFormat** As %String (VALUELIST = ,Radio,Option,Dropdown,UserEntry) [InitialExpression = "Radio"; Required;]
"Radio" - creates radio buttons (default). "Option" - used for multi-select Questions. "Dropdown" - this shows the Text of the MCAnswer or the UserEntry. If the UserEntry text is selected by the student, then a User Entry box will appear on the CSP page. "UserEntry" - use this when there is only a single User Entry text input box used in the question
• property **DisplayVertical** As %Boolean [InitialExpression = 1;]
when this is true, the MCAnswer Options are displayed with line breaks between each. Otherwise, they are displayed on a running line
• relationship **Entry** As Feedback.Question [Inverse = Slots; Cardinality = one;]
The Survey (One relationship) in which the accociated Entry is placed
• property **InstructorSpecific** As %Boolean [InitialExpression = 0;]
Indicates if this question is specifically asking for feedback on a specific instructor. If this is set to true, then the responses will be tagged to that only the instructor and his manager and the administrator will be able to view the answers to this question
• property **LinesVisible** As %Integer [InitialExpression = 1;]
this property applies only when AnswerFormat=Dropdown. This field is coded into the HTML to indicate the number of lines to show in the SELECT field.
• property **MaxAnswers** As %Integer [InitialExpression = 1; Required;]
The maximum number of answers which can be selected by the Student. If this is 1, then any AnswerFormat can be selected. If this is more than 1, then only the "Option" or "Dropdown" type of AnswerFormat can be selected.
• property **Required** As %Boolean [InitialExpression = 0;]
Boolean Property - Indicating that this Question must be filled in in order for the survey to be submitted

• property **ShowCorrectAnswer** As %Boolean [InitialExpression = 0;]
Boolean Property: True - if the survey is one where correct answers are shown, then this question will have its proper answer shown when the answers are given
• property **ShowMaxLimit** As %Boolean [InitialExpression = 1;]
when the question is made visible, if this boolean field is set to True, then a small reminder will be printed showing the MaxAnswers than can be picked
• list property **VisibleTo** As Feedback.Person
This is a list which points to people, and gets populated by the survey creator, who chooses who gets to see the collected responses to this question used in this survey

Methods

• method **RenderHTML(SessionID As %String = "", ViewerID As %String = "", breaks As Integer = 2)** returns nothing.
• method **RenderResponseHTML(SessionID, ViewerID)** returns nothing.
this is an instance method which creates the HTML output used to display this question in the survey N.B.: When breaks=0, it is assumed that any text entry will be performed by the calling

Queries

• query **SlotList()**
This is a query used to find all Question slots used in the database

Indices

• index **EntryIndex** on Entry

Feedback.Question

persistent class **Feedback.Question** extends Feedback.Entry
SQL Table Name:
Question Object is the basic build block of a survey. The Question is what is presented to the student, and the Question Object is related to the various possible answers which a student may choose (including a fill-in-the-black).

Properties

• relationship **AnswerOptions** As Feedback.AnswerOption [Inverse = Question; Cardinality = children;]

Children of Question. the different options which can be used to populate a Question. I.E. The student will choose from these options to answer the question.

• relationship **Answers** As Feedback.Answer [Inverse = Question; Cardinality = children;]

Children of Question - the Answers created by students who answered this Question as part of a survey

• property **CorrectAnswer** As Feedback.AnswerOption

(Optional) This is a pointer to the correct answer to the question. Obviously, this only applies to content-based questions, and not to feedback questions.

• relationship **Slots** As Feedback.QSlot [Inverse = Entry; Cardinality = many;]

is a Many relationship to an object which associates this Entry with a specific survey in which it is supposed to appear.

• property **UserEntryIndex** As %Integer [Calculated;]

this is a calculated property which loops through all AnswerOptions associated with this question, and returns the index if one of them is a UserEntryText object, otherwise returns 0

Methods

• method **CompressOptionPositions()** returns %Status

This is an instance method which goes through the AnswerOptions assigned to this survey, and removes any holes in their OptionOrder numbering

---

Feedback.RatingQuestion

persistent class **Feedback.RatingQuestion** extends Feedback.Question
SQL Table Name:
This is a question which merely asks the student to give a rating answer, which will be an integer ranging from MinRating to MaxRating. Automatically, MaxAnswers will be set to 1

Properties

• property **MaxRating** As %Integer [InitialExpression = 10;]

maximum integer, default = 10

• property **MinRating** As %Integer [InitialExpression = 1;]

minimum rating possible for this question, default = 1

Feedback.Registration

persistent class **Feedback.Registration** extends %Persistent

SQL Table Name:
This is a class to enable the Many-to-Many relationship between Students and Courses. Specifically: - 1 Studented can be Registered for many Courses - 1 Course can have many Students Registered for it.

Properties

• property **Cost** As %Numeric
    this is the cost of registering for this Course
• relationship **Course** As Feedback.Course [Inverse = Registrations; Cardinality = one;]
    Course for which the associated Student has Registered
• property **DateRegistered** As %Date (FORMAT = 5)
    this is a field that self-populates the first time the record is saved, to show when registration occured
• property **Paid** As %String (VALUELIST = ,Yes,No,On-Site)
    Used for courses which require payment (corporate training, etc)
• property **SelfRegistered** As %Boolean
    This is a boolean which is set to true when the student self-registers for a course (as opposed to being added from the Course Workspace)
• relationship **Student** As Feedback.Student [Inverse = Registrations; Cardinality = one;]
    The Student who is registered for the associated Course
• property **TimeRegistered** As %Time
    this is a field that self-populates the first time the record is saved, to show when registration occured

Indices

• index **CourseIndex** on Course
• index **StudentIndex** on Student

Feedback.Response

persistent class **Feedback.Response** extends %Persistent
SQL Table Name:
Child of Survey. A Response object represents the response of a single person to a particular survey. Its children are a collection of Answer objects, each of which contains a spefic QuestionID and the associated answer to that question

Properties

• relationship **Answers** As Feedback.Answer [Inverse = Response; Cardinality = many;]

One to Many relationship - a Response is related to all of the Answers given in the completion of a Survey by a specific Student

• property **DateCompleted** As %Date (FORMAT = 5)

this is the date which the survey was completed. This property is automatically populated when the response object is saved for the first time (in the %OnBeforeSave method)

• relationship **FeedbackRequest** As Feedback.FeedbackRequest [Inverse = Responses; Cardinality = one;]

This is a relationship to identify the request which instigated this response

• relationship **Respondent** As Feedback.Person [Inverse = Responses; Cardinality = one;]

This will be populated by a cookie, unless the user decides to take the survey anonymously.

• property **TimeCompleted** As %Time

This property hold the time at which this response object was completed. It is autopopulated when the object is saved for the first time (i.e. in the %OnBeforeSave method).

Queries

• query **AnonRespsByFRID(FeedbackRequestID As %String)**

used in ResponseList - returns all responses which are anonymous for a given FeedbackRequestID

• query **NonAnonRespsByFRID(FeedbackRequestID As %String)**

used in ResponseList - returns all responses which are not anonymous, and (TODO:not self-registered) for a given FeedbackRequestID

• query **NonAnonRespsByFRIDChron(FeedbackRequestID As %String)**

used in ResponseList - returns all responses (CHRONOLOGICALLY) which are not anonymous, and (TODO:not self-registered) for a given FeedbackRequestID

• query **NumberRepondentsByFRID(FeedbackRequestID As %String)**

given a FeedbackRequestID, this query returns a single column (PersonCount) which has the number of distinct respondents associated with that ID

• query **NumberReponsesByPersonID(FeedbackRequestID As %String, PersonID As %String)**

• query **ResponsesByFeedbackRequestID(FeedbackRequestID As %String)**

• query **ResponsesBySurveyID(SurveyID As %String)**

• query **SelfRegRespsByFRID(FeedbackRequestID As %String)**

TODO IMPLEMENT: used in ResponseList - returns all responses which are not anonymous, and ARE self-registered for a given FeedbackRequestID

Indices

• index **FeedbackRequestIndex** on FeedbackRequest
• index **RespondentIndex** on Respondent

Feedback.Role

persistent class **Feedback.Role** extends %Persistent
SQL Table Name:
This is an superclass from which each of the roles of a person (or 'hats') inherits. This class should be extended by any new roles added into the system, and no objects of this type should be created (it is not abstract since that would prevent general references to the Role class)

Properties

• property **PersonalInfo** As Feedback.Person [Required;]
This is a reference back to the Person who is fulfilling this role. In effect, this will create a one-to-one relationship (which is not directly supported by Cache at this time). Care must be exercised to delete any pertinant roles when a person is deletes, and to erase a reference to a role object from within its referencing person if the role is deleted.

Feedback.Session

persistent class **Feedback.Session** extends %Persistent
SQL Table Name:
A Session Object represents a discrete portion of a Course. The Session is used to break up a Course into smaller pieces about which the Instructor can request feedback from the Students. The way in which a Course is broken up is completely arbitrary, and up to the Instructor. An instructor may choose to have Sessions represent discrete windows in time (e.g Mon, Tue, Wed), or they could represent logical sections of the Course (Chapter 1, Chapter 2, etc). It is intended that the Instructor have maximum flexibilitiy in choosing how to divide the course material and time

Properties

• property **Cancelled** As %Boolean
this is a method used to insert a person (user) based on their position (role) into a Attending or Absence list (list) calling this method will loop through all of the Teaching Assignment relationships from the Course parent of Session, and checks to see

if that instructor is already in the InstructorsAttending list or InstructorsAbsent list. If it is in neither place, then the UserName is addes to the InstructorsAttending list Boolean field used to indicate that the session has been canceled.

• relationship **Course** As Feedback.Course (XMLREFERENCE = 100) [Inverse = Sessions; Cardinality = parent;]

Parent of Session - the Course is the entire content offering of which the Sessions is one subdivision (either division by time or material)

• property **Description** As %String (MAXLEN = 500)

This is a text field which the instructor can fill in for their own tracking purposes to keep track of the material covered by this course.

• relationship **EmailAnnouncements** As Feedback.EmailAnnouncement [Inverse = Session; Cardinality = children;]

Children of Session - every session can have as many email announcements as it would like, which are sent to the students (and optionally to the instructors as well)

• property **EndTime** As %Time

(Time the Session Ends)

• relationship **FeedbackRequests** As Feedback.FeedbackRequest [Inverse = Session; Cardinality = many;]

One to Many relationship - pointing to on objects which creates an accociation between this Session an a specific Survey that is to be used in the session

• list property **InstructorsAbsent** As Feedback.Person

List of Instructors (based on UserName) who are related to the parent course, but not involved in this session.

• property **SessionDate** As %Date (FORMAT = 5)

Date the session is given

• property **StartTime** As %Time

Time the session starts (optional)

• list property **StudentsAbsent** As Feedback.Person

List of Students (based on UserName) who are related to the parent course, but not attending this session.

• property **Title** As %String (MAXLEN = 250)

Title of the Session

Methods

• method **Cancel(EmailInform As %Boolean = False, Body As %String)** returns %Status

sets the Cancelled field to true. If "EmailInform = True", the method sends emails to the StudentList and InstructorList informing them that the class is canceled, where the Body argument is the text of

the email (without any argument, there is a default message crafted saying that the session for such a time and such a date has been canceled, contact the instructor for more details).

• method **ChangeAbsentStatus(PersonId, role, NewStatus)** returns %Status
This method will change the status of a user to Absent or NotAbsent, based on the PersonID passed, the role passed (Student or Instructor) and the NewStatus (Absent of NotAbsent)

Queries

• query **SessionList()**
• query **SessionListByPerson(PersonID)**

Feedback.SingleEntryQuestion

persistent class **Feedback.SingleEntryQuestion** extends Feedback.Question
SQL Table Name:
This is a question which has a single UserEntry box (which is created automatically when a question of this type is created.) MaxAnswers is set to 1, and AnswerFormat is set to "userentry"

Feedback.Slot

persistent class **Feedback.Slot** extends %Persistent
SQL Table Name:
The Slot object facilitates the Many-to-Many relationship between Surveys and Entries. The Slot Object brings together a SurveyID with an Entry (either a Label or Querstion), and also records the page on which the Entry is supposed to live (in the survey) and the Location on the page (specified by a numeric order - 1st location, 5th location, etc). Intuitively, a Survey is comprised of a number of Slots, each of which contains an Entry. Or, Questions and Labels appear in a numerous Slots on various surveys.

Properties

• property **FontSize** As %Integer (MAXVAL = 7,MINVAL = 1) [InitialExpression = 3;]
HTML Size of the the Text in the label (values must be 1-7)
• property **FontType** As %String
the type of font to display in this font.
• property **Page** As %Integer
Indicates the Survey Page on which the associated Entry is located
• property **Position** As %Integer

Indicates an integer value which represents the order on the survey page in which this entry is supposed to appear. The entries are placed in increasing order and placed on the page accordingly.

• relationship **Survey** As Feedback.Survey [Inverse = Slots; Cardinality = one;]

Methods

• method **InsertAfter(pos As %Integer)** returns %Status
  This method sets the position of the current Slot object. It takes 1 parameter (pos), sets ..Position to pos+1, and adds 1 to the Position values of all othe Slots related to this Survey NOTE: ..Survey must be set for this to work - if it is not, then this method will return a zero. If this is to be inserted into the first Position on the page, then pass 0 as the parameter
• method **MoveDown()** returns %Status
  This method moves the slot down one position. NOTE: ..Survey must be set for this to work - if it is not, then this method will return a zero. If this is already the first slot, then there will be no change
• method **MoveUp()** returns %Status
  This method moves the slot up one position. NOTE: ..Survey must be set for this to work - if it is not, then this method will return a zero. If this is already the first slot, then there will be no change
• method **SetNextPosition()** returns %Status
  This method looks at all Slot objects related to this ..Survey, and finds the maximum Slot value for all objects with a page value of ..Page (it sets ..Page=1 if it is previously not set). Then it sets ..Location to the greatest location value + 1. If the code runs without issues, a 1 is returned, otherwise a 0 is returned Note: ..Survey must be set prior to callling this method

Queries

• query **SlotsForSurvey(surveyID As %Integer)**

Indices

• index **PositionIndex** on Position [Type = bitmap;]
• index **SurveyIndex** on Survey

Feedback.Student

persistent class **Feedback.Student** extends Feedback.Role

SQL Table Name:
a Student (role) object is an individual who is being presented with the Learning Material in one way or another. The students are the Objects who will out Surveys to give feedback to the Instructors on the Material. (if the situation arrises where instructors are creating feedback for an Author, then the Instructors should take on the role of students

Properties

• relationship **Registrations** As Feedback.Registration [Inverse = Student; Cardinality = many;]
    One to Many relationship where a student has several Registrations, each of which points to an indivudual Course which the student will or has taken
• property **eLearningSubscription** As eLearningSubscription
    property pointing to the serial class which contains the properties and methods specific to the InterSystems eLearning program

Methods

• method **IsRegisteredFor(CourseId)** returns Feedback.Registration
    This method received one parameter (CourseId), and checks whether or not any of the registrations associated with this student are connected to the course with the passed ID. If so, then the RegistrationID is returned, otherwise "" is returned

---

Feedback.Subject

persistent class **Feedback.Subject** extends %Persistent
SQL Table Name:
The Subject object is the highest level of Learning Object distinction. Is defines boundaries on the Learning Object, i.e. it represents a certain set of materials. Examples of a "Subject" might be Physics 101, of U.S. History 202. I.E. it is a defined set of material which is intended to be presented over some span of time.

Properties

• relationship **AuthorCredits** As Feedback.AuthorCredit [Inverse = Subject; Cardinality = many;]
    One to Many relationship listing the objects with point to the Authors who are due credit for contributing to the Subject in one way or another (the Author list will receive access to the feedback made viewable to them by the instructors
• relationship **Courses** As Feedback.Course [Inverse = Subject; Cardinality = children;]

Parent-Child relationship. The Courses are time bound offerings of the material defined for the Subject (It makes no sense for Courses to exist without the context of a Subject (material taught in the course))

• property **Description** As %String (MAXLEN = 500)
Title of the subject - this should in some way reflect the body of material covered in the associated teaching objectives of this Subject

• relationship **RelevantTo** As Feedback.SubjectAssociation [Inverse = Subject; Cardinality = many;]

• property **SubjectCode** As %String
This will be used to create all Course Numbers which are used in the database

• property **Title** As %String (MAXLEN = 250) [Required;]
Title of the subject - this should in some way reflect the body of material covered in the associated teaching objectives of this Subject

Queries

• query **SubjectList()**
This is a list of all the Subjects in the System

• query **SubjectListByPerson(PersonID)**
This is a list of all the Subjects in the System related to PersonID

Feedback.SubjectAssociation

persistent class **Feedback.SubjectAssociation** extends %Persistent
SQL Table Name:
this is a many-to-many proxy class used to tie together people and subjects with which they are associated

Properties

• relationship **Person** As Feedback.Person [Inverse = SubjectAssociations; Cardinality = one;]
this is the person to whom this proxy class relates

• relationship **Subject** As Feedback.Subject [Inverse = RelevantTo; Cardinality = one;]
this is the subject tied to this proxy class

Indices

• index **PersonIndex** on Person

• index **SubjectIndex** on Subject

Feedback.SuperUser

persistent class **Feedback.SuperUser** extends Feedback.Role
SQL Table Name:
The SuperUser role gives the highest level of control to person, for system-wide controls

---

Feedback.Survey

persistent class **Feedback.Survey** extends %Persistent
SQL Table Name:
Parent of Response. This is the basic object for design feedback content, and containing feedback data. The Survey object is created by the instructor, and then is accessed by the students, for whom Response objects are created and related back to the original survey.

Properties

• property **CreatedBy** As Feedback.Person
this stores the PersonID of the creator of this survey
• property **Description** As %String (MAXLEN = 500)
This stores a description for the survey being used
• relationship **RelevantTo** As Feedback.SurveyAssociation [Inverse = Survey; Cardinality = many;]
• relationship **Slots** As Feedback.Slot [Inverse = Survey; Cardinality = many;]
One to Many relationship - the Slots relationship points to an object which associates this survey with a specific entry that belongs in the survey
• property **SurveyName** As %String (MAXLEN = 250) [Required;]
this is a name given to the survey for selection purposes
• relationship **UsedBy** As Feedback.FeedbackRequest [Inverse = Survey; Cardinality = many;]
One to Many relationship - this pointsw to an object which creates an association between this Survey, and all of the Sessions in which it is Used.

Methods

• method **CompressSlotPositions()** returns %Status
This is an instance method which goes through the slots assigned to this survey, and removes any holes in their position numbering

Queries

• query **SurveyList()**
    This is a query which finds all surveys in the database so they can be displayed
• query **SurveyListByPerson(PersonID)**
    This is a query used to show all surveys Associated with a certain Person

---

Feedback.SurveyAssociation

persistent class **Feedback.SurveyAssociation** extends %Persistent
SQL Table Name:
This is a many-to-many proxy class which is used to tie people to surveys which are relevant to them.

Properties

• relationship **Person** As Feedback.Person [Inverse = SurveyAssociations; Cardinality = one;]
    this refers to one person object, which is tied to the survey
• relationship **Survey** As Feedback.Survey [Inverse = RelevantTo; Cardinality = one;]
    This relationship points to a survey object

Indices

• index **PersonIndex** on Person
• index **SurveyIndex** on Survey

Feedback.TeachingAssignment

persistent class **Feedback.TeachingAssignment** extends %Persistent
SQL Table Name:
Class created to enable many-to-many interaction between Instructors and Courses. This means: 1 Instructor can Teach many Courses 1 Course can have many Instructors scheduled to Teach

Properties

• relationship **Course** As Feedback.Course [Inverse = TeachingAssignments; Cardinality = one;]

The Course assigned to the given Instructor to teach

• relationship **Instructor** As Feedback.Instructor [Inverse = TeachingAssignments; Cardinality = one;]

The Instuctor assigned to teach the Course associated with this object

Indices

• index **CourseIndex** on Course
• index **InstructorIndex** on Instructor

Feedback.TrueFalse

persistent class **Feedback.TrueFalse** extends Feedback.Question
SQL Table Name:
This is a True or False question, which automatically creates MCAnswer objects for True and False, and sets MaxAnswer to 1.

Feedback.UserEntryText

persistent class **Feedback.UserEntryText** extends Feedback.AnswerOption
SQL Table Name:
Extends AnswerOption. A user entry blank in which the student types whatever they want

Properties

• property **MaxLength** As %Integer [InitialExpression = 256;]

Maximum length of the user entry form field, initial value = 256

• property **NonMCOption** As %Boolean [InitialExpression = 0;]

This is a flag which is set the UserEntryText Option is not supposed to be a multiple-choice option, instead it is meant to accompany the multiple-choice options. E.G. "Please Choose One of the Following" [5 Radio options] Please Explain [Text Box]

• property **NumCols** As %Integer [InitialExpression = 50;]

The number of Columns to create for the UserEntry box on the CSP form

• property **NumRows** As %Integer [InitialExpression = 1;]

The number of Rows to create for the UserEntry box on the CSP form

Queries

• query **FindAll()**
This is a query to find all UserEntryText objects

Feedback.YesNo

persistent class **Feedback.YesNo** extends Feedback.Question
SQL Table Name:
This is a Yes or No question, which automatically creates MCAnswer objects for Yes and No, and sets MaxAnswer to 1.

Appendix D – CSP Pages

AddCourseToContract.csp – InterSystems specific popup page used to associate a Course with a Contract

AddEntryUse.csp – popup window called from within SurveyWorkspace.csp that allows the user to create new slots or pull data from existing objects in the database; a parameter is passed to this page indicating whether the context is for questions or labels

AddSurveyUse.csp – popup window from SessionWorkspace.csp that allows a Survey object to be added to a Session

AdminTools.csp – brings together the HTML of TopMenu.csp, TopAdminMenu.csp and UserBar.csp

ChangeCourseTitle.csp – popup window used for changing the title of a Course; launched from CourseWorkspace.csp

ChangeSessionTitle.csp – popup window used for changing the title of a Session; launched from SessionWorkspace.csp

ChangeSubjectTitle.csp – popup window used for changing the title and description of a Subject; launched from SubjectWorkspace.csp

ChangeSurveyName.csp – popup window used for changing the name of a Survey; launched from SurveyWorkspace.csp

CourseWorkspace.csp – main page used for manipulating Course objects and managing the Session objects that it contains

CreateCourse.csp – jump page for working with Courses; from here a user can choose to create a new Course or edit an existing Course

CreateSession.csp – former jump page for working with Sessions; now the navigation goes through the CourseWorkspace.csp page

CreateSubject.csp – jump page for working with Subjects; from here a user can choose to create a new Subject or edit an existing Subject

CreateSurvey.csp – jump page for working with Surveys; from here a user can choose to create a new Survey or edit an existing Survey

EditCompany.csp – InterSystems page used for editing customer Company information

EditCourse.csp – container page holding CourseWorkspace.csp and two frames of ListPeopleInCourse.csp (once for Instructors and once for Students)

EditEmail.csp – popup window that allows an Email object to be edited.

EditPerson.csp – page that allows a Person object to be edited; a user must be an administrator to have access to all fields

EditProfile.csp – container page for profile alterations of users of the OnFORME backend; contains TopMenu.csp, UserBar.csp and EditProfileContent.csp

EditProfileContent.csp – page on which the user can change their personal information, password, etc

EditRegistration.csp – InterSystems page used to edit a student's registration information and add appropriate billing details

EditRespondentProfile.csp – container page for respondent profile content; contains RespondentBar.csp and EditProfileContent.csp

EditSession.csp – container page for editing Sessions; contains SessionWorkspace.csp and ParticipationList.csp

EditSlot.csp – popup container page that hold SlotProperties.csp and ViewSlot.csp; this page is accessed from SurveyWorkspace.csp

EditSubject.csp – container page for editing Subjects; contains SubjectWorkspace.csp and ListAssociations.csp

EditSurvey.csp – container page holding TopMenuandUser.csp and SurveyWorkspace

EditSurveyUse.csp – popup page that allows the properties of a FeedbackRequest object to be controlled; this popup is called from SurveyWorkspace.csp

Forbidden.csp – when a user tries to access a page for which they do not have a high enough access level, they will be redirected here

Goodbye.csp – due to the intricacies of using the Preserve flag in the Survey Workspace, an external page needed to be used to turn off the preserve setting following the user leaving that area; this page is a popup to clean-up the preserved session

Index.csp – the homepage of OnFORME, displaying system announcements

ISCTools.csp – container page that combines TopMenu.csp, TopISCMenu.csp and UserBar.csp

ListAssociations.csp – page that lists all of the people associated with a subject; gives the option of adding or removing associations

ListFeedbackRequests.csp – use of this page has been discontinued, but at one point it had been used as a jump page to different feedback requests and their responses

ListPeopleinCourse.csp – used in EditCourse.csp to list the people registered in a Course; a passed parameter signifies whether Instructors are displayed or Students are displayed

ListSlots.csp – discontinued page; used to be used to add Slots to a Survey

Login.csp – the page through which all users must first come in order to enter their login information

NewRespondentProfile.csp – when respondents are given the option of registering themselves and they do not yet have a profile in the system, they are brought to this page to create a username and password

OnSiteContract.csp – InterSystems page that allows them to edit all of the information pertaining to an On-Site training contact

ParticipationList.csp – lists all of the registered instructors and students in a session, and allows their status to be changed to indicate that they were absent from that session

PersonPopup.csp – popup window that gives a quick view of the data for a person in the database; allows navigation to the edit page for that person

PreviewSurvey.csp – this is a popup window that renders the survey, as the respondent will see it

RespondentBar.csp – this is the menu bar included in a survey for the respondent to use; it includes the option of changing their profile information, logging out or reporting a bug

RespondentFooter.inc – this is an include file that holds the HTML that serves a the footer in all of the pages that a respondent sees

RespondentHeader.inc – this is an include file that holds the HTML that serves a the header in all of the pages that a respondent sees

ResponseList.csp – reporting page that lists the responses and links to view the date in each response

ResponseSet.csp – reporting page that dumps all of the data from a feedback request into a single table so it can be copied into a spreadsheet for further evaluation

ResponseSummary.csp – reporting page that shows the executive summary of the responses for a feedback request; this page is accessed from a course or a session object

SendingEmail.csp – popup window that asks the user to wait while the email is being sent

SessionWorkspace.csp

ShowSurvey.csp – respondent page; this is the gateway to a feedback request; respondents hit this page with a parameter with signifies which feedback request they are responding to

SlotProperties.csp – page that allows all of the properties of a slot object to be controlled: either a question slot or a label slot and the data that it holds

SubjectAssociations.csp – an administrative page that allows people to be associated (given rights to) various Subject objects within the system; when a user has access to a Subject, they by default have access to the Courses and Sessions that belong to the Subject

SubjectWorkspace.csp

SubmitSurvey.csp – page that collects the data submitted when the respondent hits "Submit" on the ShowSurvey.csp page; stores the Answers in the database and then displays the thank you text or redirects the user

SurveyAssociations.csp – an administrative page that allows people to be associated (given rights to) various Survey objects within the system

SurveyWorkspace.csp – page in which Survey objects are manipulated, and their slots controlled; this page is unusual because it preserves a state on the server in order to handle the intricacies of manipulating Surveys and their sub-objects

SystemStatistics.csp – page that shows all of the FeedbackRequest objects in the database and their statistics; this is only viewable to superusers

TopAdminMenu.csp – menu navigation bar for administrative tool pages

TopISCMenu.csp – menu navigation bar for InterSystems tool pages

TopMenu.csp – contains the navigation logic for OnFORME, allowing users to jump to different parts of the system

TopMenuandUser.csp – brings together TopMenu.csp and UserBar.csp

UserBar.csp – contains the user option links including edit profile, logout, and report bug

ViewEmail.csp – popup window that shows the details of an email that was sent by OnFORME to the people in a Course

ViewResponse.csp – a popup page that renders a survey and populates it with the response given by a user

ViewSlot.csp – preview of a slot in a survey

Help/BugReport.csp – page explaining how to report a bug with OnFORME

Help/CreateCourse.csp – help tutorial explaining the steps for creating a course

Help/CreateSession.csp – help tutorial explaining the steps for creating a session

Help/CreateSubject.csp – help tutorial explaining the steps for creating a subject

Help/CreateSurvey.csp – help tutorial explaining the steps for creating a survey

Help/FAQ.csp – a few Frequently Asked Questions concerning the use of the OnFORME system

Help/Glossary.csp – a glossary of OnFORME terminology

Help/Help.csp – the help homepage showing the different help options

Help/Overview.csp – an OnFORME system overview, explaining how the various pieces fit together

Help/Tutorials.csp – the jump page listing the different tutorials

Help/UsingSurvey.csp – help tutorial explaining the steps for using a survey in a session

Appendix E – Competitive Analysis Findings

The following lists qualitative data collected on applications included in the competitive analysis. A table showing the data used for the quantitative analysis follows this.

OnFORME Application Background:	
Created By	Ben Spead
System Provider Web Address	http://i2i.mit.edu/feedback
Contact Person	Ben Spead
Contact Information	speadbf1@mit.edu
Intended User Base	Educational Environments
Total Length of Development Cycle	- 10 months total at 45% effort - 4.5 months of full-time effort
Number of People on Development Team	1
Initial Startup Cost (License)	$0
Annual Cost (License)	$0
Authentication Model	UserName / Password
User Roles	Student, Instructor, Author Manager, Administrator, SuperUser
Additional Noteworthy Features	

QTools Application Background:	
Created By	MIT- AMPS
System Provider Web Address	http://amps-tools.mit.edu/qtools/
Contact Person	Mark Brown
Contact Information	mwbrown@mit.edu
Intended User Base	MIT Community
Total Length of Development Cycle	- 1 year at 25% effort - 3 months of combined effort (full time whole team)
Number of People on Development Team	4 (proj mgr, GUI designer, Graphic Designer, Programmer)
Initial Startup Cost (License)	$0
Annual Cost (License)	- Semester Subscription or 1 time cost (e.g. 1 time survey with 150 people ~ $250) - Complicated price structure (based on expected maintenance cost) - 1 semester, 50 students ~ $300-$500 (?? - not cost modeled out)
Authentication Model	Kerberos, UserName / Password
User Roles	Administrator (various levels of rights - read access, write access, etc) (Instructor), User (Student)
Additional Noteworthy Features	- automated Survey Invite - Kerberos authentication - "Integer Only" type Answer Options, allowing for more automatic statistical analysis - invitation list is based on email list - when you set properties of survey, you have the option of setting certificates, then it is a global option (can't include non-MIT) - once any responses have been collected, nothing can be changed in functionality

Navigo Application Background:	
Created By	Indiana University Stanford University
System Provider Web Address	http://navigo.sourceforge.net
Contact Person	Lance Speelmon, UI
Contact Information	lance@iu.edu, 317-407-5359
Intended User Base	Educational Environments - Graded Assessments
Total Length of Development Cycle	- April - Nov. '03 10 full time (4 dev, 2 arch, 1 mgr, 3 grphc) - Nov '03 - May '04, 10 above + 6 more Stan. Devs
Number of People on Development Team	10 (then) - 16 (now)
Initial Startup Cost (License)	$0
Annual Cost (License)	$0
Authentication Model	UserName / Password
User Roles	Student, Instructor, TA, Grader, Faculty of Record, Administrative
Additional Noteworthy Features	- based on OKI standards - will release many parts of design in light-weight form, with the intent that they can be replaced by other OKI solutions - full release set for July '05 - profs can grade assignments while having the student's ID masked - question types include file upload and audio answer - questions use a WYSIWYG editor for profs - used QTI 1.2 as standard to avoid having to model question types.

Blackboard Application Background:	
Created By	Blackboard Co.
System Provider Web Address	http://www.blackboard.com/
Contact Person	
Contact Information	
Intended User Base	Educational Environments
Total Length of Development Cycle	
Number of People on Development Team	
Initial Startup Cost (License)	
Annual Cost (License)	$7500 for basic institution license
Authentication Model	UserName / Password
User Roles	Student, TA, Instructor
Additional Noteworthy Features	- option for MathML and equations in the question text - survey time limit - "Forced Complete" option - for 1 question per page, possibility of preventing backtracking

SloanSpace Application Background:	
Created By	
System Provider Web Address	
Contact Person	
Contact Information	
Intended User Base	Educational Environments
Total Length of Development Cycle	
Number of People on Development Team	
Initial Startup Cost (License)	
Annual Cost (License)	
Authentication Model	UserName / Password
User Roles	Student, TA, Instructor
Additional Noteworthy Features	- date question option - attachment question option - email notification - can send email to those who have not yet responded

FAST Application Background:	
Created By	Mount Royal College
System Provider Web Address	http://www.getfast.ca
Contact Person	Dr. Bruce Ravelli Zvjezdan Patz
Contact Information	(403) 240-7716 bravelli@mtroyal.ab.ca zpatz@shaw.ca
Intended User Base	Educational Environments
Total Length of Development Cycle	- 5 years total cycle time - 8-12 months full-time effort
Number of People on Development Team	2 (developer and architect)
Initial Startup Cost (License)	$5000 + Cold Fusion License
Annual Cost (License)	$1000 for upgrades
Authentication Model	Anonymous (password authenticatoin)
User Roles	Student, Instructor
Additional Noteworthy Features	- Allow students to view survey results - auto chart generation - creation of excel file w/results - discussion board for instructors using the tool

QuizLab Application Background:	
Created By	Teacher Vision @ Pearson Education
System Provider Web Address	http://www.quizlab.com
Contact Person	Vince Krist
Contact Information	(702) 614-1719 Vince.Krist@fen.com
Intended User Base	Educational Environments
Total Length of Development Cycle	2.5 months
Number of People on Development Team	1
Initial Startup Cost (License)	$0
Annual Cost (License)	$33 per user
Authentication Model	Random Password
User Roles	Instructor, Student, Parent
Additional Noteworthy Features	- integrated with MyGradebook.com - Quiz Packs with existing quizzes can be purchased - built with Cold Fusion

Flashlight Online Application Background:	
Created By	Washington State University
System Provider Web Address	http://flashlightonline.wsu.edu
Contact Person	Steven Saltzberg Sr. Consultant Robin Zuniga
Contact Information	saltzberg@tltgroup.org (804) 360-1440 zuniga@tltgroup.org 512-428-1042
Intended User Base	Educational Environments
Total Length of Development Cycle	2 years full time
Number of People on Development Team	5
Initial Startup Cost (License)	$0
Annual Cost (License)	$4000-$5000 including 2 days of consulting time $2500 a year without the consulting time
Authentication Model	Username / Password
User Roles	Author, Respondents, Administrator (creates accounts, groups, etc)
Additional Noteworthy Features	- access to database of 500 Survey Templates - best practices guide to Survey creation and distribution - access to peer reviewed surveys - old surveys can be archived to take them off of the screen

	OnFORME	QTools	Navigo	Blackboard Survey Manager	SloanSpace Surveys	FAST	QuizLab	Flashlight Online
Application Attributes:								
Educational Environment Data Model	Yes	Yes	Yes	Yes	Yes	Limited	Yes	No
Direct Student URL Access	Yes	Yes	Yes	No	No	Yes	No	Yes
Local Hosting Option	Yes	No	Yes	Yes	Yes	Yes	No	No
Integrated User Database including all User Types	Yes	No	Yes	Yes	Yes	No	No	No
Student Registrations for Course	Yes	No	Yes	Yes	Yes	No	No	No
Session Based Absenteeism Control	Yes	No	No	No	No	No	No	No
Instructor Registrations for Course	Yes	No	Yes	Yes	Yes	No	No	Limited
Data Organization and Access:								
Survey Reusability	Yes	No	Process	No	No	No	Yes	No
Survey Reuse Presentation Parameters	Yes	No	Process	No	No	No	No	No
Survey Cloning	Yes	Yes	Process	Limited	Yes	Yes	Yes	Yes
Question Reusability	Yes	No	Process	No	No	No	No	No
Question Reuse Presentation Parameters	Yes	No	No	No	No	No	No	No
Question Cloning	Yes	Yes	Process	Limited	Yes	Yes	Yes	No
Author Based Survey Deletion Rights	Yes	No	Yes	Yes	Yes	No	Yes	Yes
Login Based Survey Viewing	Yes	No	Process	No	No	No	Yes	Yes
Login Based Subject Viewing	Yes	Yes	No	Yes	Yes	Yes	Yes	No
Respondent Access Control:								
Single Submission Control	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Editable Submission	Process	Yes	Yes	No	Yes	No	No	No
Multiple Submission Option	Yes	Yes	Yes	Yes	Yes	No	Yes	No
Respondent Login	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Anonymous Responses	Yes	Yes	Yes	No	No	Yes	No	Yes
Anonymous Option	Yes	No	No	No	No	No	No	No
Verification Login	Yes	Yes	No	No	No	No	No	No
Verification Option	Yes	No	No	No	No	No	No	No
Student Self-Registration Option	Yes	Yes	No	No	Yes	No	No	No
Attendance Based Survey Access	Yes	No	No	No	No	No	No	No
Active Survey Period	Yes	Yes	Yes	Yes	No	No	Yes	No

	OnFORME	QTools	Navigo	Blackboard Survey Manager	SloanSpace Surveys	FAST	QuizLab	Flashlight Online
Question Types:								
Radio Question Type	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Radio with Textbox Option Question Type	Yes	No	No	No	No	No	No	No
Radio with Comment Question Type	Yes	No	Yes	No	No	No	No	No
Checkbox (Multi-Select) Questions	Yes	Yes	Yes	Yes	Yes	No	No	Yes
Checkbox with Textbox Option Question Type	Yes	No	No	No	No	No	No	No
Checkbox with Comment Question Type	Yes	No	Yes	No	No	No	No	No
Dropdown Question Type	Yes	Yes	No	No	Yes	No	No	No
Dropdown with Textbox Option Question Type	Yes	No	No	No	No	No	No	No
Dropdown with Comment Question Type	Yes	No	No	No	No	No	No	No
Multi-Select Dropdown Question Type	Yes	No	No	No	No	No	No	No
Free Text Question Type	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Ranking Question	No	No	No	Yes	No	No	No	No
Question Attributes:								
Answer Alignment Option (V/H)	Yes	No	No	No	Yes	No	No	No
Textbox Height Control	Yes	Yes	No	No	Yes	No	No	No
Textbox Size Control	Yes	Yes	No	No	Limited	No	No	No
Font Control	Yes	No	No	No	Yes	No	Yes	No
Required Option	Yes	No	No	No	Yes	No	No	No
Max Answer Limit (multi-select)	Yes	No	No	No	No	No	No	No
Max Answer Limit Display Option	Yes	No	No	No	No	No	No	No
Question Grading (with "Right" answers)	Process	No	Yes	No	No	No	Yes	No
Option to Show Respondent "Right" answers	Process	No	Yes	No	No	No	Yes	No
Conditional Question Branching	No	No	No	No	No	No	No	No
Instructor Specific Option	Yes	No	No	No	No	No	No	No
Feedback Channel to Material Authors	Process	No	No	No	No	No	No	No

	OnFORME	QTools	Navigo	Blackboard Survey Manager	SloanSpace Surveys	FAST	QuizLab	Flashlight Online
Survey Presentation:								
Custom HTML Wrapper for Respondent Pages	Yes	Yes	Yes	No	No	No	No	No
Label Objects	Yes	Yes	Yes	No	No	No	No	No
Customizable Thank You	Yes	Yes	Yes	No	No	No	Yes	No
Post Survey Redirect	Yes	Yes	Yes	No	No	No	No	Yes
Multi-page Surveys	Process	No	Yes	Limted	No	No	Yes	No
Communication Tools:								
Email Functionality	Yes	Yes	Process	Yes	Yes	Limited	Yes	No
Email Queuing	Yes	No	No	No	Yes	No	No	No
Registration-Based Emails	Yes	No	No	Yes	Yes	No	Yes	No
Attendance-Based Emails	Yes	No	No	No	No	No	No	No
Rule-Triggered Emails	Process	No	Yes	No	No	No	No	No
Other:								
Cross-Browser Compatibility	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Does Not Require Client-Side JRE	Process	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Question Limit	none	none	none	none	none	20	200	none

Appendix F – System Test Results

Site	Course	Instructors	Survey Name	Start Date	End Date	Submission Mode	Login Mode	Associate Response	Allow Self-Registration	Total Reponses	Anononymous Responses	Total Repondents	Associated Respondents	Anonymous Repondents	Population Size	Response Rate
Beta	ESD.801		2003 ESD.801 Leadership Weekend			S	Y	Y	N	29	0	29	29	0	45	64%
Beta	ESD.801	Cutcher-Gershenfeld,Joel	ESD.801 Leadership Development Feedback Survey			S	Y	Y	N	14	0	14	14	0	45	31%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 1 -- Feb 4th	2/9	2/22	S	Y	Y	Y	35	0	35	35	0	35	100%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 2 and 3 -- Feb 9th and 11th	2/11	2/22	S	Y	Y	Y	33	0	33	33	0	35	94%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 4 and 5 -- Feb 16th and 18th	2/20	2/28	S	Y	Y	Y	30	0	30	30	0	34	88%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 6-7 on Feb 23 and 25	2/26	3/3	M	Y	Y	Y	31	0	31	31	0	34	91%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 8-9 on Mar 1 and 3	3/4	3/12	S	Y	Y	N	27	0	27	27	0	34	79%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 10-11 on Mar 8 and 10	3/8	3/20	S	Y	Y	Y	31	0	31	31	0	34	91%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 12 and 13 -- March 15th and 17th	3/17	4/1	S	Y	Y	N	28	0	28	28	0	34	82%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 14 and 15 -- March 29th and 31st	4/1	4/21	S	Y	Y	N	30	0	30	30	0	34	88%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 16 and 17 -- April 5th and 7th	4/10	4/21	S	Y	Y	N	28	0	28	28	1	34	85%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 18 and 19 -- April 12th and 14th	4/16	4/26	S	Y	Y	N	31	0	31	31	0	34	91%

Site	Course	Instructors	Survey Name	Start Date	End Date	Submission Mode	Login Mode	Associate Response	Allow Self-Registration	Total Reponses	Anononymous Responses	Total Repondents	Associated Respondents	Anonymous Repondents	Population Size	Response Rate
I2I	ESD.140	Cutcher-Gershenfeld, Joel	ESD.140 Sessions 20 -- April 21st	4/20	4/30	S	Y	Y	N	28	0	28	28	0	34	82%
I2I	ESD.140	Cutcher-Gershenfeld, Joel	Auto Industry System Study Survey	4/20	4/30	S	Y	Y	N	16	0	16	16	0	34	47%
I2I	ESD.80	Clark, Joel Kirchain, Randolph	ESD.80 Critiques		6/1	M	Y	Y	Y	57	0	32	32	0	32	100%
I2I	CU 5CMI3	Ash, John Nuttall, Bill	5CMI3 Evaluation			S	A	Y	N	5	5	0	0	0	8	63%
I2I	CU MOTI	Oliver, Nick	Review of MOTI Consulting Project and Assignment			S	A	Y	N	16	16	0	0	0	69	23%
ISC	CORE1	Solon, Joel (3 day delay)	Original Learning Services Survey	3/18	4/25	S	Y	SC	N	5	0	5	5	0	11	45%
ISC	CUSTOM2	Solon, Joel (11 day delay)	Original Learning Services Survey	3/12	4/25	S	Y	SC	N	0	0	0	0	0	7	0%
ISC	CORE4	McDaniel, Adele (7 day delay)	Original Learning Services Survey	3/22	4/25	S	Y	SC	N	3	1	3	2	1	10	30%
ISC	CORE5	McDaniel, Adele (6 day delay)	Original Learning Services Survey	3/22	4/25	S	Y	SC	N	3	1	3	2	1	10	30%
ISC	SYS6	O'Leary, Sue	Original Learning Services Survey	4/6	4/10	S	Y	SC	Y	12	2	12	10	2	12	100%
ISC	SYS16	O'Leary, Sue (3 day delay)	Original Learning Services Survey	4/1	5/3	S	Y	SC	Y	5	0	5	5	0	8	63%
ISC	SYS20	O'Leary, Sue	Original Learning Services Survey	4/19	4/20	S	Y	SC	Y	4	2	4	2	2	4	100%
ISC	CORE19	McDaniel, Adele	Original Learning Services Survey	4/12	4/18	S	Y	SC	N	12	2	11	9	2	15	80%
ISC	CSP26	Solon, Joel (2 day delay)	Original Learning Services Survey	4/7	5/3	S	Y	SC	N	1	0	1	1	0	10	10%

Appendix G – How to Access an OnFORME Demo

The OnFORME System has been deployed at the Engineering Systems Learning Center of MIT, and is available for demo.

Viewing a Demonstration Survey

To see a demonstration survey, go to:
http://i2i.mit.edu/feedback/ShowSurvey.csp?FeedbackRequestID=7
(NOTE: the above URL is case sensitive)

Three options are available for filling out the survey:

1. Logging in under the Demo Profile:
   Username: **Demo**
   Password: **feedback**

2. Taking the survey Anonymously

3. Registering and taking the survey (this will create a new user profile in the system).

Fill in the survey, and hit submit.

Seeing the OnFORME Application

To access the back-end of the system, go to:
http://i2i.mit.edu/feedback/

Log into the system using the Demo Profile:
Username: **Demo**
Password: **feedback**

The Demo profile is set up with Instructor privileges, which allow surveys, sessions and courses to be created and manipulated. To view the demo survey, click on the "Survey Workspace" tab in the top menu, and then click "Edit" next to the Demonstration Survey. Feel free to experiment with the functionality and appearance of the survey, but you will not be able to save it as it is used for demonstration. To see how the survey is used for the demo Feedback Request accessed, click on the "Course Workspace", and select the Demonstration Course. There Select the Demo Session. Here, the Use characteristics can be viewed, as well as the Responses collected with this Feedback Request. To view the responses, click on the link to the right of the survey listed (there is some sample data which can be viewed in the response pages).

Further details of the use of the system can be found in the help files (the link is in the upper right corner of the browser).

Appendix H – Installation Instructions

This appendix contains a rough step-by-step guide to installing Caché and then installing the OnFORME system. The directions are for a Windows 2000 machine running IIS, but can easily be adapted for any operating system supported by the Caché database.

NOTE: This represents a best effort rough guide to installation. For detailed instructions for the latest version of Caché, please see the Caché documentation. This appendix is not supposed to be a replacement for the Caché reference materials concerning system installation and configuration. Additionally, the details on the setup of OnFORME are meant to be a rough guide as well, as alterations may be necessary to tailor the system to the needs of the specific user population.

I. Installation of Caché
    A. Install Caché, choosing default installation options, with the exception of:
        1. Select "Custom" for the Setup type
        2. Select "Unicode"
        3. Choose all components except for "Weblink"

II. Configuring Caché
    A. Click on the blue "Caché Cube" in the lower right toolbar
    B. Select the "Configuration Manager"
        1. Check "Start Caché on Boot"
        2. Set the memory for the Database Cache to be 200 MB
        3. Set the memory for the Routine Cache to 50 MB
    C. If the license key was not manually typed in during installation, and a key license file exists, place that file in "Program Files\Cache\Mgr\"
        1. Caché needs to be restarted after the key is added

III. Configuring OnFORME Database
    A. In the Configuration Manager, click the "Namespaces" tab
    B. "Add" a Namespace and name it (e.g. "ONFORME")
    C. Select "Define New Database"
    D. Create a Database for that namespace and name it (e.g. "ONFORME")
    E. Pick the database location
        1. it is recommended that the location not be within the \Cachesys directory, as that directory is overwritten on Caché upgrades
    F. Select the initial size of the Database – recommended 20 MB
    G. Select Caché Std, 8 KB
    H. Click "Finish"

IV. Configure OnFORME CSP Application
    A. Open the "Configuration Manager"
    B. Click on the "CSP" tab
    C. Expand the "Applications" List
        1. Right Click on the "CSP\<new namespace>" application and "Remove" it
    D. Right click on the "Applications" list and select "Add"
    E. Name URL "/feedback"

1. NOTE: At the time of writing, parts of the OnFORME code relied upon this exact application name being used. If a different application is to be used, then changes will need to be made to the code to make it work

F. Select the namespace to be the namespace created above (e.g. "ONFORME")

G. Creating a directory

1. Decide on a location for the Web content (CSP pages and images)
2. Using windows explorer, create a "\feedback" directory in that location

H. Set "Caché Physical Path" to be the full path of the directory that you just created. (e.g. "C:\MyCaché\CSP\ONFORME\feedback\")

I. Increase the "Default Timeout" – e.g. to 36000

1. This field determines the session timeout for people logged into OnFORME pages. A longer timeout is preferred so that people will not be cut off if they are editing a survey. However, this value should be tuned to the specific needs of the installation site.

J. Set the Default SuperPage to be "Feedback.CSPSuperPage" (this is case sensitive)

K. Select the "Advanced Tab" in the Configuration Manager

1. Expand the "SQL" List
    a) Change "Support Delimited Identifiers" to "Yes"

V. Installing OnFORME Code

A. Click on the Caché cube and select "Studio"

B. Select "File->Change Namespace" and select the new namespace that was recently created

C. Select "Tools->Import Local"

D. Find the OnFORME export XML file (note, this is the "code" and can be downloaded from SourceForge.net)

E. Confirm that each of the CSP pages has the "\feedback\" in front of them (if they don't then the CSP Application was not set up correctly).

F. Import and compile the code

1. If there were errors during compile, try a "Build->Rebuild All".
2. If you get an error that says "'X' is an SQL reserved word…"
    a) Go to Config. Manager->Advanced->SQL
    b) Change "Support Delimited Identifiers" to "No"
    c) Click "OK" and Activate
    d) Now go back, change the value to "Yes"
    e) Click "OK" then recompile – it should work.
3. If the above doesn't work, then you need to figure out what is wrong with you configuration, and you will need to be able to compile the code before proceeding

G. Save the Project in Studio

H. Copy the "Images" directory into the directory holding the CSP files (e.g. "…\CSP\ONFORME\feedback\"

VI. Configuring OnFORME

A. Click on the Caché cube and select "terminal"

B. Change to your new namespace by typing:

1. zn “<namespacename>”

C. Now the namespace should be shown in the prompt. Type:
    1. do ^InitializeFeedbackSystem

D. Enter the requested information – don’t forget the super-user login data!

VII. Configuring IIS

This is optional – if you don’t have IIS then you can use the lightweight web server that comes with Caché. However, this is strongly recommended against because it has not been thoroughly tested for use with OnFORME. Is it recommended that a real web server (IIS, Apache, etc) be used with OnFORME.

A. Open the IIS manager GUI

B. Right-click on the web server, and select “New Virtual Directory”
    1. Name the application “\feedback” (note – this is currently required due to the use of this string in the code, so if a different application name is required, the code will have to be adjusted).
    2. Point to the directory holding the CSP files for OnFORME “..\feedback\”
    3. Check the “Execute” box

VIII. Testing the system

A. Point a web browser to http://localhost/feedback/index.csp
    1. If IIS is not being used, then point to http://localhost:1972/feedback/index.csp

B. A login page should appear, type in the SuperUser information

C. This should bring you to the OnFORME application.

Good Luck!!