

# Requirements Document and Design Rationale

## UCC course submission system upgrade

Authors: Sabar Banks  
Ronishia Kears  
Bastian Tenbergen

Team International

Date created: 10/15/2007  
Current Version: 12/05/2007

Table Of contents:

PART I – Introduction	3
1.1 Purpose	3
1.2 Document Conventions	3
1.3 Intended Audience and Reading Suggestions	3
1.4 Project Scope	3
1.5 References	3
PART 2 – Requirements Specification	4
2.1 Product Perspective	4
2.2 Preliminary Assumptions & Claims	4
2.3 User Classes and Requirements	4
2.3.1 Persona: UCC Chair	5
2.3.2 Persona: UCC Member	5
2.3.3 Persona: Department Secretary	5
2.4 Technical Requirements and Limitations	5
2.5 Interface Design Choices	6
3. System Features	7
4. User Needs Analysis	8
5. External Interface Requirements	11
5.1 User Interface	11
5.2 Hardware and Software Interfaces	11
5.3 Communication Interfaces	11
PART 3 – Design Rationale	12
6. Prototype Evolution / Project Lifecycle	12
6.1 Low-Fidelity Prototype: The Website Approach	12
6.2 Prototype Iteration: The Applet Approach	14
6.3 Final Design: A Standalone application with server backed	15
7. Conclusion	17
Appendix: Use Cases	18

## **1. Introduction**

### **1.1 Purpose**

The purpose of this project is to create a robust and usable product that meets the client's needs. Additionally this project is aimed at making the task of submitting new and revised courses easier. The project is designed to make a way that the client can save and reviewed before submitting the final copy. This project will be created to be safe and secure for every user.

### **1.2 Document Conventions**

While creating this document, every defined requirement and system feature has its own level of priority.

### **1.3 Intended Audience and Reading Suggestions**

This document was made with the intent of modeling a system for course submissions and reviews. With this in mind the target audience is mainly any member of the UCC and academic staff who wish to submit a course proposal.

### **1.4 Project Scope**

The scope of the revision to the UCC course submission system spans across four main points. The first point allows for any user the ability to save their progress at any time. Second, the text areas on any form must be word wrapping, or for in layman's terms, all text fits in the size of the text area. Third is providing UCC staff with all the relevant contact information for a reply upon approval or denial. Last is the system should have the ability to be self updating; all courses that are created are to be recorded automatically into a master document.

### **1.5 References**

- UCC Homepage:
  - o [www.oswego.edu/UCC](http://www.oswego.edu/UCC)

## **2. Requirements Specifications**

### **2.1 Product Perspective**

The Course Submission and Tracking System of the Undergraduate Curriculum Council at SUNY Oswego that is currently being used, is far away from being a user-friendly framework that allows for quick procedures, follow-ups or status tracking. The system that is proposed in this document is intended to be a replacement for the existing framework. It is designed to surmount the limitations the members of the UCC are faced with right now and shall stream-line the process of submitting new courses and support the decision of course submission reviews. Due to the fact that the system will be used by single users with different roles and responsibilities in contrast to user groups with similar requirements to the system, the proposed framework needs to be robust against different views of similar datasets, depending on the requirements of a specific user.

### **2.2 Preliminary Assumptions & Claims**

In order to assess the user requirements better, a number of assumptions and claims need to be considered. Verbalizing these considerations will help in focusing on the user experience during the design of the data representation and – more importantly – during the design of the interface.

This design proposal assumes that the current system that SUNY Oswego's UCC uses is insufficient to allow for a stream-lined processing of new course submissions. It is claimed that a framework that understands the roles and responsibilities of each UCC member will aid the users to perform more efficiently and effectively. We furthermore claim that a unified system that offers a different representation of the same data for different users is capable of providing the functionality that affords efficiency and effectiveness. It is assumed that a fully automated system is preferred over paper-based approaches and we claim that web-based interface is the ideal solution.

### **2.3 User Classes and Requirements**

The Undergraduate Curriculum Council is a board of experts that understand the responsibility of providing the student body with a curriculum that affords a well-rounded education as well as the freedom to choose what course offerings fits best in the individual fields of interest. In order to make ideal informed decisions, the UCC evaluates new course submissions and assesses how well the newly submitted course fits into the context of the curriculum. The UCC consists of faculty representatives of different schools and departments that represent each sub-division. The UCC chair is a single person that maintains the overview over the entire process and assigns course submissions to different UCC members. All UCC members as well as the UCC chair are elected to serve on the board for two years. This requires the system to be robust against a variety of different knowledge levels of the submission and approval process of the users that will be using the framework. In addition, it is typically the secretaries of different

departments that submit the courses. Department secretaries are typically hired for a longer time period than two years. They do not have an in-depth knowledge of the UCC course submission approval process and are not experts on database systems. Their interaction with the system is only limited, since they will primarily use the system to submit new courses, update existing courses or update previous course submissions. The below personas will help in focusing on the tasks and abilities of different user classes to afford a user-centered design process.

### **2.3.1 Persona: UCC Chair**

Henry is a 42 year old college professor with a PhD in Marketing. It is the end of his second year as the chair of UCC and he is very familiar with the process of approving new course submissions. His role as UCC chair makes it necessary for him to know which UCC member is working on which submission and has to maintain an administrative overview over the members and their roles. Soon, he will have to transfer his responsibilities to the next UCC chair. He uses computers on a daily basis, but the works behind a computer program are like pure magic to him. As long as his computer and the software work, he is happy.

### **2.3.2 Persona: UCC Member**

Martha is a 58 year old faculty member in the Department of Psychology. She has been working in her field for almost 30 years and just recently became accustomed to using a computer for her statistical analyses and teaching efforts. She is somewhat familiar with the course submission approval procedure in the UCC as it is the beginning of her first year on the council. She has been guided through a mostly paper-based process once and has no clue how a computer program will help her in doing her job on UCC.

### **2.3.3 Persona: Department Secretary**

Maria is a 31 year old department secretary in the Music Department. She has in-depth knowledge of different administrative procedures and uses a computer every day to fulfill her responsibilities. She has a limited understanding of how to maintain (excel spread-sheet) databases and update datasets and does not really want to know how to set up or query a SQL database. Although she has already submitted numerous courses to the UCC and worked closely with its members, she has no idea about the approval process. She typically uses the UCC submission interface once or twice a semester.

## **2.4 Technical Requirements and Limitations**

Just as important as the user requirements are the technical requirements and limitations. These immediately impact on the design choices and potentially restrict certain interaction types.

The most prominent complaint about the existing framework is that new course submissions arrive at the UCC in the form of unformatted text in emails. In order to allow for easy usability of the data, the course submissions shall be delivered in a form that is easier to integrate into the overall design flow.

Another important requirement is that roles, names and identities of UCC members must be easily changeable. The system must be easily adaptable to changes made to the list of UCC members.

The system must be maintainable with a minimum of technical expertise and shall allow for easy adaptability and extensibility.

Security is one of the most important technical requirements. The system shall only be accessible by authorized people and shall eliminate the ability to manipulate data in a destructive way.

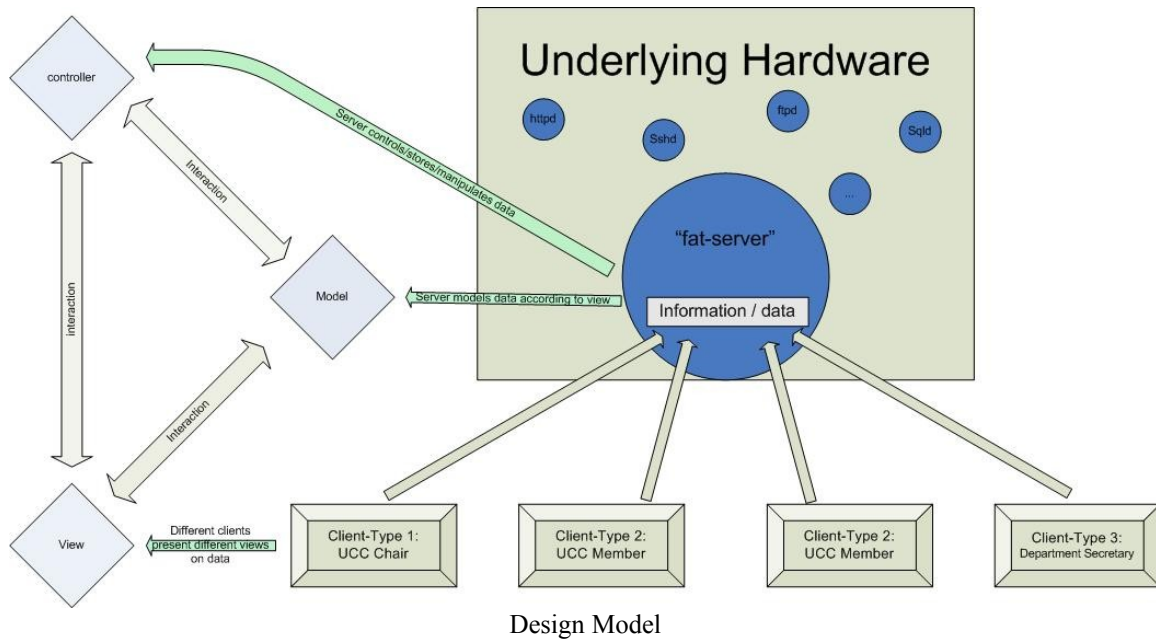
## **2.5 Interface Design Choices**

The interaction of users with the proposed framework must be as versatile as possible, due to the number of different users with different responsibilities and tasks on the same data. Therefore, a hybrid interaction style is the ideal solution. The interface will incorporate a small amount of exploratory interaction possibilities as well as conversing interaction types in form of “wizards”. Also, the interface will incorporate instructive interaction aspects in critical functions that require safe data handling. In order to represent the data sets as user friendly as possible, and in order to fulfill the most important technical requirement of easy maintenance and extensibility, the system will be strictly designed according to the Model-View-Controller (MVC) architecture. The “model” being the underlying that users have to work with, the “view” being the interface that represents the model and the “controller” being a underlying production server that handles user communication and data manipulation. We choose a so-called “fat server” design to implement the underlying controller-logic. The “view” will be implemented using web-applets that display different interface flavors that represent only those data that a particular user needs. Depending on the user that uses the system, a certain interface representation will be chosen that allows the user to quickly perform the necessary task without having to worry about interface options that are only available for a different user. This is intended to lower the interface complexity and therefore presenting less opportunity for frustration.

The data itself will be represented in abstract object form. Being closely connected to the principle of object-oriented programming, the data will be presented in objects that belong together. A super-object may contain sub-objects. The user can therefore traverse the data either breadth-first, by browsing through data sets (objects) of the same kind or depth-first by choosing a specific object-type first and then browsing through the data. For example, each course submission will be an object that contains or is associated with a UCC member that is currently working on this submission (which can be a UCC-member-object), a course-object that contains information on the course, and a number of associated information like dates, status information, etc. This carefully developed conceptual model of regarding data as objects will aid novice users in understanding the course submission approval procedure as well as give an easy understandable interaction concept with the interface.

To give a better understanding of the interaction concept of our interface framework, please refer to the below diagram of the design model.

Image 1:



### 3. System Features

Here lists the major intended functions of the revisions to the UCC course submission system.

#### 3.1 Password Protection of course submission database

- Priority: high.
- Details of the login implementation can be seen Use Case #1. Essentially, no person should be able to access any software intended to be used only by the UCC and the OSU academic staff.

#### 3.2 Course proposal system allows for saving at anytime

- Priority: high
- Details regarding the ability to save at anytime can be seen in Use Case #1. Essentially any person submitting any form of a course submission can be saved and resumed at anytime.
- All proposals are to be formatted in a clear manner.

#### 3.3 System self updates

- Priority: high
- Requirements for this feature to be invoked are as simple as submitting, approving, or deleting a course or proposal.

- Functional requirement: TBD.

### 3.4 **System can export any course or proposal as file**

- Priority: High
- Functional requirements: TBD

### 3.5 **System must have help full tool tips**

- Priority: medium
- Functional requirements: TBD.

### 3.6 **Wizard will be implemented for the novice user**

- Priority: medium.
- Functional requirements: TBD.

## 4. **User Needs Analysis**

Outside the above mentioned system requirements, it is important to establish the user needs on a task-oriented level. Since the project team has only limited ability to interact with the target user groups, the most effective needs and requirements establishing procedures (e.g. Contextual inquiry, user interviews, Focus Group Sessions, etc) are not feasible. Instead, an expert interview was conducted in order to establish a detailed description of the tasks of a typical user from each user group. The results can be seen in the below task diagrams.

### 4.1 **Hierarchical Task Analyses**

The following Hierarchical Task Analyses (HTA) depict an average most popular task for a typical member of every user group. Please note, that these HTAs are not sensitive to the interface type or process structure. As the UCC currently uses a mainly paper-based approach, the below HTA are valid for paper-based procedures as well as the software interface this project is aiming at producing.

#### 4.1.1 **UCC Chair – Typical Task: Assign Course Submissions to UCC members**

0. in order to assign a new course submission to a UCC member
  1. log on to the system
    - 1.1 enter user name into log in screen
    - 1.2 enter password into log in screen
    - 1.3 click submit button
  2. acquire knowledge about current statuses of course submissions
    - 2.1 scan list of all submissions in progress
    - 2.2 identify new submission without assigned UCC member
  3. update knowledge about work load of every UCC members
    - 3.1 scan list of all UCC members

- 3.2 a) identify UCC representative that represents department of newly entered course submission
- b) identify UCC member with least work load should the UCC department representative have a too high workload already
4. assign course to identified UCC member
5. repeat steps 2 – 4 for every unassigned course submission
6. log off from the system

#### 4.1.2 UCC Member – Typical Task: Review Course submission

0. in order to review a course submission
1. log on to the system
  - 1.1 enter user name into log in screen
  - 1.2 enter password into log in screen
  - 1.3 click submit button
2. acquire knowledge about current statuses of course submissions
  - 2.1 scan list of all submissions in progress
  - 2.2 choose submission to work on
3. review submission:  
in random order, do the following:
  - 3.1 ensure, pre-requisites do not cascade
  - 3.2 ensure, pre-requisites exist and are suitable
  - 3.3 verify correct course number
  - 3.4 ensure correct and easy-to-understand catalog descriptions
  - 3.5 correct typos

in case of errors/mistakes in any of the above steps, contact the person that submitted the course, retrieve updated information and iterate.
4. mark the course as reviewed
5. log off from the system
6. meet with other UCC members and vote on course submission approval

#### 4.1.3 Course Submitting Department Rep – Typical Task: Submit Course/Update Course

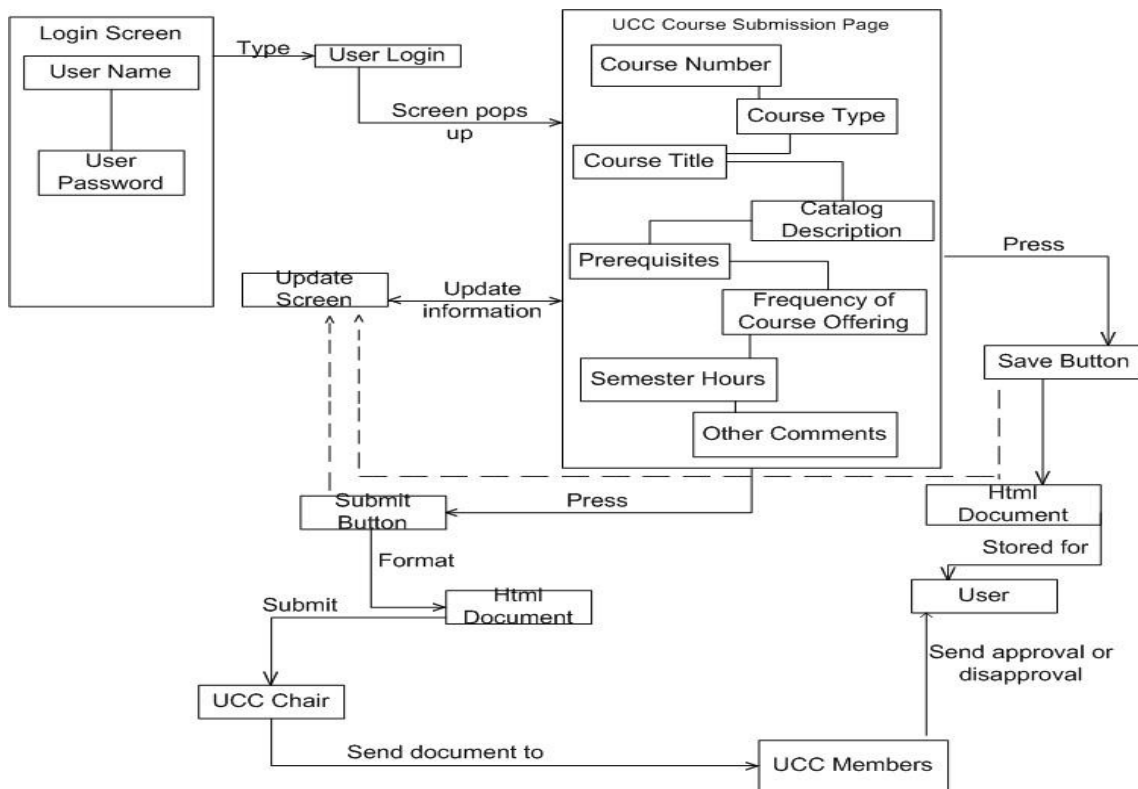
0. in order to submit a a new course or update an existing submission
1. log on to the system
  - 6.1 enter user name into log in screen
  - 6.2 enter password into log in screen
  - 6.3 click submit button
7. acquire knowledge about current statuses of course submissions
  - 7.1 scan list of all submissions in progress
  - 7.2 choose submission to update
  - 7.3 review submission:
    - 7.3.1. correct any errors depicted by the UCC member that reviews this submissions
    - 7.3.2. retrieve further information, if needed

- 7.3.3. update existing information as needed
- 7.4 submit a new course
  - 7.4.1. open course submission interface
  - 7.4.2. fill out necessary fields
  - 7.4.3. check for typos and mistakes
  - 7.4.4. click submit button
- 8. log off from the system

## 4.2 System State Diagram

Since this project's main goal is to develop a software interface that aids all members of UCC and stakeholders in carrying out their tasks, it is of high importance to design a system that incorporates a stream-lined process. A theoretical system interaction prototype that was developed on the basis of the HTA to accomplish that task is outlined in the following System State Diagram.

Image 2:



System State Diagram

## **5. External Interface Requirements.**

### **5.1 User Interfaces**

The User Interface is designed to make the client job easier. All UIs are designed to take input from the client and display it into a well formatted html document. This document will make it easier for the reader to read.

Furthermore the UI will supply the client a save button for their personal use, if they are not ready to submit a final copy. In addition, the model, for this project, UI will supply a way for the client to update the courses for other viewers. Clients will be allowed to receive a copy of the final draft as a confirmation that it has been received.

### **5.2 Hardware & Software Interfaces**

The hardware that will be key figures in this project would be the keyboard, mouse, and printer. Since the hardware on helps to manipulate data in a database, complex hardware constructs are not needed.

In addition to the main hardware components is the software relationship that goes long with them. As the input software, the mouse and keyboard, the software will be smart enough to analyze user input on the fly, as it is coming in. The software will be geared towards making decisions about informing the user about other options and warnings. All of which would be based on the user input as it is being entered.

What is to be needed to support the new UCC system would be various small software components. These small components would be the Java 6 runtime environment, Firefox (or some kind of web browser)

### **5.3 Communication Interfaces**

Communication is something that has been adequately provided by the SUNY Oswego emailing system itself. For this project, it uses the CTS emailing service instead of implementing a separate emailing client. More specifically, if any client desires to email a course proposal, rejection, or approval, etc... The CTS email service will be loaded, via web, and all proper information will be passed to the CTS emailing system in the correct fields of a composed message.

The database will be modeled after a fat-server system. This would allow most of the computational tasks to be done by the server. From this increases the degree centralization. The requirements for this being that each user have a working Internet connection, C.T.S. account, and the privileges to access the database.

## 6. Prototype Evolution / Project Lifecycle

In order to implement the project identified above, a RAD/JAD Lifecycle model has been chosen. RAD/JAD models (Rapid Application Development with Joint Application Development Meetings) have proven to yield great success in implementing software projects with a small to medium number of team members. JAD workshops have been conducted throughout the span of the project to establish and verify the previously conceived requirements. Furthermore, throughout the implementation phase, informal evaluations and assessments of the current prototype have been made to ensure a user centered design and an efficient fulfilling of the user requirements.

In the following, the most important prototype iterations are listed. Along with screenshots of the design of the interface, an explanation is given what critical adjustments have been made to improve the interface.

### 6.1 Low-Fidelity Prototype: The Website Approach

The Low-Fidelity Prototype was the first sketch in preparing a user interface that accommodates the previously established requirements. Special attention has been given to the idea of ease of maintenance and high usability. A design similar to a website has been chosen, as it is assumed that the users of our system are all likely to be familiar with using the Internet to find information and submitting information using web-forms. To lower the learning curve, a design similar to the front-end of the online course management system “Angel” has been adopted, as the institution that this project is designed for, recently adopted “Angel” as the online course management system, and faculty and staff of the institution are already familiar with the interface.

Aside from a decreased learning curve, designing the interface as a website has the advantage that changes to the interface can be made centrally from the server, without running into versioning conflicts when different users are using outdated versions of the interface. Furthermore, websites are available on demand, without having to install software on local machines and without having to prepare the end-user machine to comply with the software. This furthermore increases the usability and decreases the need for specially trained support personnel and maintenance.

Despite these advantages, a website design has been discarded as the final solution for the interface. Main factors for this decision were critical problems in the usability of such an interface and security problems. With a freely available website that is hosted on a public server, user authentication and data safety is a core requirement. As both is not possible without the need of a database server that must be managed by personnel, specially trained for database administration and UNIX user administration, this design choice is not feasible for this project. In addition, there are some critical usability issues regarding the course management software “Angel”, that in some cases lead to a decreased user experience and inhibited usability. Identifying these issues, however, is beyond the scope of this project and this document.

Image 3:

### Log-in Page

Log-in

user : [ ]

password : [ ]

State University of New York at Oswego, Oswego, NY 13126-3599

### Activity Page

Home Page

Hello

Activity

The Wall

State University of New York at Oswego, Oswego, NY 13126-3599

### UCC Submission Page

Course Submission Form

Scroll panel ↓

State University of New York at Oswego, Oswego, NY 13126-3599

## 6.2 Prototype Iteration: The Applet Approach

Designing an Applet as the main interface for the system was a second, more important milestone in the design process. As applets run from a web server on the local machine of the user, they combine the advantages of ease of maintenance of websites with streamlined design abilities (such as widget placement) and security features of applications. As they are running from a centralized server as well, no installation on the client computer is necessary. As Image 4 depicts below, the Applet Approach was very easy to design specifically for the task at hand.

As major disadvantage to the Applet Approach was that due to the Sandbox security framework that Sun conceived for Java applets (i.e. applets cannot access local files without signing the applet first, applets cannot access the network to other computers than the host machine). Although securing the applet from unauthorized use is easily possible with a background light-weight Java-based authentication server that is contacted by the applet to log the current user in, one major requirement would not be able to have been fulfilled without third party software. Since it was a critical user-centered non-functional requirement to automatically spell-check and correct what the user is writing, the Applet Approach has been discarded. Since applets can only connect to the host computers via a network connection, and the spell-checking feature requires a steady connection to Google's Web Services (which cannot be relayed through a server-program on the host computer due to other technical reasons), the Applet Approach was not considered feasible for this project.

Image 4:



Applet-Prototype

### 6.3 Final Design: A Standalone application with server backed.

The various prototype iterations have made obvious where design limitations are and how feasible it is to implement the needs and requirements into a specific design. Although implementing the system in form of a standalone Java-Swing-based application was initially considered to be too complex to maintain, it still presents itself as the ideal solution for the wide range of requirements that the system must fulfill besides ease of maintenance. Also, the work that went into designing the interface in the Applet Approach could easily be adapted to satisfy a standalone application. Widget-placement, color settings and interaction scheme were completely taken over from the previous prototype iterations and yielded even better results when not seen in the context of a website. Therefore, the final design of the UCC course submission tracking system meets all previously established constraints. Please refer to images 5 and 6.

Although the system has been taken out of the context of a website, the design is carefully chosen to resemble such, which is assumed to decrease the learning curve in interacting with the design. As established at the beginning of the project, all users are assumed to have fundamental knowledge about how to interact with a website and can transpose this knowledge immediately into the system that this project suggests.

Image 5:



UCC Activity Page

The UCC Activity Page is the main page of the submission tracking system and does not only allow for immediate access to various UCC-related resources, it also facilitates ease of communication between the UCC members through the Message Center and the Wall. This approach resembles the current layout of the UCC website, which also decreases the learning curve.

Security and maintenance is made especially easy in this design. Through a light-weight background Java-based server, users can authenticate themselves so that only authorized users can use the system. Furthermore, user authentication allows for a specialized view on the available data (i.e. submitted courses) depending on the role of the member that is logged in. The back end server also handles updates of the user interface, which comprises communication between users, updates of the data-sets and software updates of the interface software. The end user does not have to be concerned with that. Since both the interface software as well as the server are entirely written in Java, the framework is robust enough to allow for two critical maintenance problems to be solved: no special hardware or software platforms are needed to run the software and software maintenance and adaptation can be completed without the need of specially trained personnel – for instance: any student employee majoring in Computer Science is able to do that.

Correct Formatting of the submission document is automatically taken care of. The end user does not have to be concerned with formatting errors, duplicate entries (of course numbers, for instance) or falsely entered course submissions. Also, due to integrated spell-checking ability, user input is automatically asynchronously corrected.

Image 6:

The screenshot shows a web browser window displaying the 'UCC Course Submission' form. At the top, there is a green header with the OSWEGO State University of New York logo and the title 'UCC Course Submission'. Below the header, there are two buttons: 'Home' and 'Back'. The form itself is divided into several sections:

- Composed By:** A text input field.
- Department Chair:** A text input field.
- Department Chair Email:** A text input field.
- Additional Contact 1:** A text input field.
- Additional Contact 2:** A text input field.
- Course Number:** A section with the instruction 'Assign a course prefix and number appropriate to the course content.' and a text input field.
- Course Type:** A section with the instruction 'Specify if the course is new or an update to an existing course.' and two radio buttons: 'New Course Number' and 'Updated/Revised Course Content (same number)'.
- Course Title:** A section with the instruction 'Assign a course title that briefly and adequately identifies the course content.' and a text input field.
- Catalogue Description:** A section with the instruction 'Write a concise course description (preferably 50 words or less) for the college catal...'. The text is truncated.

The bottom of the screenshot shows the Windows taskbar with various open applications like 'Faulty by Design', 'Inbox - Thunderbird', 'todo.txt - Notepad', 'project', and 'Requirements Doc...'. The system clock shows '1:55 PM'.

UCC Course Submission Page

## 7. Conclusion

Although the various prototype iterations cause the final system design to be slightly different from the initially conceived system state diagram, the system is able to fulfill all initially established requirements. Since the requirements have been chosen carefully, no feature had to be dropped and no requirement is missed.

Unfortunately, this project comes to an conclusion without having been formally evaluated. Usability Tests ought to be conducted to assess and potentially improve the performance of the suggested interface. An informal evaluation, however, showed that the system is meeting the intended goals and only made a few potential improvements obvious. These include added functionality (e.g. an instant messaging feature for UCC members that are currently logged in) for on-site use, an stream-lined placement of the different widgets and links and a slightly smoother color-scheme.

Future work ought to address the suggestions for improvement established by the informal evaluation. Also, future work should consist of accurate user testing and a formal evaluation as well as an iterative testing and implementation cycle that incrementally improves the program's quality given a newly established set of goals. Ideally, for this task, the end users shall closely be involved, potentially even part of the development team, as it is very hard to design a user-centered framework without having access to the users throughout the project's lifecycle.

## Appendix: Use Cases

### #1

#### Use Case: Login

Scope: Initialization, login, and view load.

Level: User.

Primary Actor: UCC Member.

Stakeholders:

- UCC Member: would like to log in to the UCC database with the appropriate GUI interface that is designed for their specific level in the UCC hierarchy. Also each UCC member would not want unauthorized individuals to have access to their restricted data.
- Professor Wenderholm: aims to obtain a workable design (at the very least) that can be implemented into replace or augment the UCC course submission system.
- Team International (working title): To develop a system for UCC that is easily maintainable; future modifications can be implemented with ease.

Pre-Conditions:

- UCC member who wished to log in must have a user name and password.
- UCC member must have software installed on their current machine.
- UCC member must have working broadband connection.

Post Conditions:

- UCC member must be logged into the system.
- UCC member must have a view of the database that is geared toward their position in the UCC hierarchy.

Main Scenario:

1. UCC member activates course submission program.
2. Course submission application initiates and connects to the Fat-server database.
3. Course submission application then requests username and password; login GUI/applet loads.
4. UCC member enters username and password.
5. Course submission application searches the faculty database and confirms.
6. Course submission database loads UCC member GUI/applet/HTML web page.
7. Course submission program informs UCC member about new messages, info, and other updates.

Alternate Scenarios:

- A. Application fails inexplicably at any time:
  - System traces error.
  - System records error.
- 1. N/A
- 2. Cannot connect to server:
  - a. Display appropriate message regarding connection issues.
  - b. Program terminates.
- 3. N/A
- 4. Invalid format:
  - a. Display appropriate message regarding the issue.
  - b. Delete current text in username and password text areas
  - c. Proceed from step 4 in the main success scenario.
- 5. Invalid username and password combination:
  - a. Number of allowed retries have not been met;
    - i. Display appropriate message regarding this issue.
    - ii. Delete entered information.
    - iii. Proceed from step 4 in the main success scenario.
  - b. Number of allowed retries have been met;
    - i. Display appropriate message regarding this issue.
    - ii. Place lock on current profile that access is being attempted on.
    - iii. Terminate program.
- 6. N/A
- 7. N/A

## #2

**Use Case:** Submitting a course proposal.

Scope: Course submission.

Level: User.

Primary Actor: UCC member/ Professor/Secretary

Stakeholders:

- UCC: upon the submission of a course proposal, the UCC would like to be informed about the individual who submitted the proposal. In addition, the UCC would like the system to be robust enough to prevent course number and name overlap.
- Professor: would like to be able to create a course proposal project, save the progress and continue at anytime in the future. Also any for that course proposals are typed into must be formatted.
- Professor Wenderholm: aims to obtain a workable design (at the very least) that can be implemented into replace or augment the UCC course submission system.
- Team International (working title): To develop a system for UCC that is easily maintainable; future modifications can be implemented with ease.

Pre-Conditions:

- Actor must have a working username and password.
- Actor must have successfully logged into the system.

Post Conditions:

- Actor must have submitted a course proposal.
- Course proposal must have a proper format and properly indicate a course proposal.
- Appropriate UCC staff must be emailed with the current UCC proposal.

Main Success:

1. Application loads to page in which the desired activity can be chosen and waits.
2. User chooses to begin a new course proposal project.
3. Application loads to a new project form.
4. User enters necessary information regarding the department chair.
5. User enters information regarding his/her faculty information.
6. User enters desired course proposal information.
7. User submits proposal.
8. Application processes information into a text file, excel file, or XML file.
9. Application then loads to the CTS email login page.
10. User inputs login information.

11. Application loads to the intended email account with an activated compose-mail draft; the subject has the course name and number. The application then waits for the user to add any extra emails.
12. User enters extra emails (if needed).
13. User sends mail.
14. Application sends email to their intended destinations.
15. Application records this activity for this account into account history.
16. Application then asks if the user would like to submit another course proposal.
17. User chooses not to submit another course proposal.
18. Application loads to the default page.

Alternate Success Scenarios:

- A) System fails inexplicably at anytime:
    - a. Display an appropriate message to the user.
    - b. Trace error.
    - c. Record error.
    - d. Terminate application.
  - B) Internet connection fails at anytime:
    - a. Application records current data that has been entered in current forms.
    - b. A message is displayed to the user indicating and Internet failure.
    - c. Application loads to the login screen.
  - C) At any time the user decides to save his/her current progress:
    - a. Application saves current progress in a well formatted file.
    - b. Application puts this course proposal into the “current projects” folder.
- 
1. n/a
  2. User chooses to be load an existing project:
    - a. Application displays all saved project from within the last thirty days.
    - b. User selects which project to load.
    - c. Application loads the saved work into the respective forms on the course proposal forms page.
  3. n/a.
  4. n/a.
  5. n/a.
  6. Invalid course information:
    - a. No match found for UCC chair or faculty information:
      - i. Application displays appropriate message to user and erases relating fields.
      - ii. User reenters correct data.
    - b. Course number already in use;
      - i. Application displays appropriate message.
      - ii. Application displays a list of possible choices related to entry.
        - User selects a choice provided;
          - a. Application enters selected course numbers.

- User declines choices provided;
    - a. Application deletes previous entry.
    - b. User enters another course number.
  - c. Course name already in use:
    - i. Application displays appropriate message regarding issue.
    - ii. Application deletes previous entry and waits.
    - iii. User enters another course number.
  - d. Course does not exist( in the event of an update):
    - i. Application informs the user that this course must be created.
17. User decided to start another project:
- a. Proceed from step 3.

Special Requirements:

- Text areas must word wrap so the text does not go off screen.
- Each text area must be easily printable.

Technology and data variations list:

- User must be a member of the UCC board or a professor and have a working I.T account.
- Any course data, submitted or approved, must be represented in excel spread sheets in addition to a text file.

Frequency of occurrence:

- Application can be used repeatedly.

Open issues:

- None.