WEB2.0 Accreditation and Departmental Management System

Requirements document and Execution plan

Michael Brooks, Owain Moss, Jonah Hirsch, Eddie Hillenbrand, Kyoko Makino

Ver 1.1

2/1/12

WEB2.0 Accreditation and Departmental Management System

Requirements document and Execution plan

# Table of Contents

Introduction…………………………………………………………………………………….2

Problem Statement…………………………………………………………………………2

Product/Solution Statement……………………………………………………………3

Requirements and Functional Specifications…………………………………..4

Constraints and Feasibility Issues…………………………………………………….8

Project Execution Plan…………………………………………………………………….9

Appendix……………………………………………………………………………………….10

2/1/2012

ABET is an accreditation organization founded in 1932 that creates standards which serves as a way to measure the experience and understanding of graduates of a certain institution. Northern Arizona University’s Computer Science program first received ABET’s accreditation on October 1996, and continues to be accredited. In order for an institution to receive ABET accreditation, the institution must create courses that meet ABET guidelines and goes through a long process of documenting all of their activities. The problem with this model hides in the process of documentation. The faculty members not only have to create the methods to assess the department’s overall performance, but also how to quantify their achievements. The objective of our project is to create a system which provides a skeleton of documentation that faculty members can fill out and use. The data from these skeletons will be able to auto-generate tools such as email notifications, schedules, reports, surveys, and/or rubrics that are easy to use to keep track of ABET accreditation procedures.

Prior to the introduction of this project, the accreditation materials were produced solely by hand. In other words, there are no tools currently available for faculty members of any institution that could produce, keep track of, and submit materials to ABET. The complexity and extent of this task comes into play when extracting each element and considering what it is composed of. Take NAU’s Computer Science program for example. One of the documents that CS faculty needs to produce is titled “Outcomes Description and Assessment Procedure.” The purpose of this document is to provide detailed objectives of what the program should provide to students. Each objective is accompanied with an outcome goal and its detailed description, the list of courses that should meet the outcome, how these outcomes are assessed, and assessment goals to achieve. Assessments are categorized as direct and indirect: Direct assessments are measured by teaching rubrics. Each rubric has description of what it measures, where and when it is measured, and its measurement processes. Indirect assessments are measured by surveys. Surveys are given to current students, alumni members, instructors, and department chairs. Each survey has its rubric that is composed of description, where and when it is measured, and its measurement processes. Faculty members are not only required to construct these guidelines, but need to evaluate the objectives, assessments, and/or goals every so often. ABET accreditation evaluations occur in intervals of seven years, so collected data must be maintained and stored properly.

As previously mentioned, currently there is no tool that provides support for this task. All objective and assessment creation, data collection, surveys, and evaluations are done by hand, on paper. Considering the volume of the task, this is not a very effective way to handle ABET accreditation process. Our system must therefore provide solutions for:

* Keeping track of current and future tasks
* Data collection and data storage
* Easy-to-understand user interface

Our approach in solving the problem is to provide users a flexible tool that they can use to create their own ABET accreditation documents and data associated with them. The system must be flexible because every institution will have different criteria and needs to fulfill their accreditation, but also be solid enough to provide necessary functionalities. By providing support for the following, the system will functional as well as flexible:

* Assessment scheduler
* E-mail Notification to faculties

The scheduler and e-mail notifications are used to keep track of current and future tasks.

* Skeleton document/form creation that contain outcome, assessment and goals

Data will be collected by institutions’ faculty members. Although we cannot provide content such as objectives, descriptions, course lists, and assessments, providing outlines for documents will certainly reduce overhead.

* Evaluation solution
* Final report creation
* Rubric creation
* Survey creation

Based on data provided, these will provide means to analyze data in a presentable ways to ABET.

Outcome

* Assessment goals

Assessment

Survey

* Description
* Addressee
* Date (next fire date & data)

Rubric

* Description
* Rating

Evaluation

* Sponsor Eval. of Capstone

Final Exam

Outcomes derive assessments, and rubrics and surveys are generated from assessments. Final exam requirements and evaluations form separate categories.

Based on our client input, our system must meet the following functional requirements:

* Authorization
* Display
* Measuring Assessment Outcomes
* Scheduling
* Event log
* Reporting
* Data

Authorization given to administrators grants him/her the ability to create or modify objects, create measures and attach them to objectives. The login system will support both Central Authentication Service (CAS) and Open Authorization (OAuth) capabilities.

**Use Case 1: System log in and object creation**

Scenario: Bob tries to log in and create an object.

Actor(s): Bob, a faculty member of A1 University’s CS department.

User Steps and system responses:

1. Bob fires the ABET ADMS.
2. Bob enters his user name and password.
3. The screen displays the user interface with create/modify object option.
4. Bob selects the ‘create object’ option.
5. Bob is able to create object.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the Object Manager (OM). The UI handles the front-end display including the workspace, buttons, and menus. The OM handles the creation of objects.

The display is used to show the listing of all ABET tasks, all objects/goals and their current status indicated by green, yellow, red lights, listing and searching capabilities, and course content such as exams that need to be reviewed by other instructors.

**Use Case 2: Listing and searching of pending ABET tasks**

Scenario: Bob wants to look through the pending ABET tasks and see who is in charge of that task.

Actor(s): Bob, a faculty member of A1 University’s CS department.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects the option to display pending ABET tasks.
3. The screen displays pending ABET tasks.
4. Bob selects ‘search’ to filter through the pending tasks to see which instructor is in charge of each task.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the Object Manager (OM). The UI handles the front-end display including the workspace, buttons and menus. The OM handles the listing of pending ABET tasks and searching capability.

Measuring assessment outcomes is used to quantify satisfied assessment goals by allowing users to take surveys and complete rubrics.

**Use Case 3: Taking survey**

Scenario: Bob wants to measure how much of his goals are met by giving his students course evaluations.

Actor(s): Bob, a faculty member of A1 University’s CS department, and CS students.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects ‘create survey’.
3. The system creates a survey.
4. Bob sends it to his students.
5. Students evaluate the course.
6. Bob receives the surveys back from students and uses them to determine whether his goals are met.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the Survey module. The UI handles the front-end display including the workspace, buttons, and menus. The survey module handles the creation of survey based on the collected data.

Scheduling functionality allows complex scheduling based on courses, semester, and season. Every instructor will have an account and receive reminders to assess Outcomes such as completing rubrics. Periodical surveys can also be scheduled through this functionality.

**Use Case 4: Scheduling a task**

Scenario: Bob wants to schedule ABET data collection on his course for the next semester.

Actor(s): Bob, a faculty member of A1 University’s CS department.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects ‘scheduler’.
3. The system displays a workspace to enter tasks to be scheduled, and menus to select a time frame.
4. Bob enters information and selects ‘confirm’ to save changes.
5. The scheduler is set for upcoming tasks.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the Scheduler. The UI handles the front-end display including the workspace, buttons, and menus. The scheduler handles the creation of future events and email notifications.

Event logging saves events related to faculties, courses, objectives, and measures.

**Use Case 5: Logging event for courses.**

Scenario: Bob enters an event for his course.

Actor(s): Bob, a faculty member of A1 University’s CS department.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects ‘event log’.
3. The system displays a workspace to enter events.
4. Bob enters information for his course and selects ‘confirm’ to save changes.
5. The event is saved for particular course.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the Event handler. The UI handles the front-end display including the workspace, buttons, and menus. The event handler handles event logs.

Reports contain logged information and status. Reports are created based on different perspectives. An example of a report request would be “show me a faculty member report”. This would summarize for a faculty member what his/her tasks are in the system, and his/her current status, such as completion ratio. Class reporting is also possible for core classes. Note that not all classes will produce class reports since not every course is mandatory to take and therefore has no measures.

**Use Case 6: Creating faculty report**

Scenario: Bob wants to create a faculty report on Chris.

Actor(s): Bob and Chris are both faculty members of A1 University’s CS department.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects ‘create faculty report’.
3. The system displays faculty members of A1 University’s CS department.
4. Bob selects ‘Chris’ and ‘report categories’.
5. The system generates a report on Chris based on the selected categories.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the Report module. The UI handles the front-end display including the workspace, buttons, and menus. The report module handles the creation of reports.

Data collection and storage functions are the core operations of the system. They also perform data backup and are used to output Office Open XML compatible documents.

**Use Case 7: Creating a word document with collected data.**

Scenario: Bob wants to create a word document with collected data.

Actor(s): Bob, a faculty member of A1 University’s CS department.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects ‘create word document’.
3. The system displays a sample word document filled with data.
4. Bob edits the document and prints it out.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the Data store module. The UI handles the front-end display including the workspace, buttons, and menus. The data store is responsible for saving, backing up, restoring, and retrieving of data.

The website should start loading immediately. The client-side HTML/CSS/JavaScript must complete loading in less than 5 seconds or display a loading message. No API call can exceed 30 seconds in execution time. The auto-shared feature of the Google Datastore should not enforce limitations on the data storage requirements for this application. Sort order optimization declarations are required for all datasets that are requested with a sort order.

There are few constraint and feasibility issues that we may encounter:

* Cost constraints

This project is projected to cost a grand total of $0.00 USD for our valued customer.

* Language constraints

Python, JavaScript, HTML, and CSS will be used in this project. This application will work with the services provided by the Google App Engine, as well as CAS and OAuth. This application shall be fully functional with any modern standards compliant browser. The web application component of this application can run on commonly used operating system such as OSX, Linux, and Windows, by means of the Google Python SDK.

* Feasibility issue

Not all of the outcome goals can be assessed in an automated fashion. There may be some outcomes that are very complex to assess and may not be completed within the given time frame. There may be aspects of the ABET report that need to be filled in manually or otherwise modified using a word processor.

# Project Execution Plan

|  |  |  |
| --- | --- | --- |
| Milestone 1 | Requirements and Execution plan document | 2/7/12 |
| Milestone 1 | Presentation 1 | 2/9/12 |
| Milestone 2 | Software design spec | 2/23/12 |
| Milestone 3 | Implementation of the core system. | TBD |
| Milestone 4 | Presentation 2 | 4/5/12 |
| Milestone 5 | Implementation of additional functionalities. | TBD |
| Milestone 6 | Testing | TBD |
| Milestone 7 | Final report | 5/7 |

# APPENDIX

<http://www.abet.org>

<http://nau.edu/CEFNS/Engineering/Computer-Science/Degrees-and-Programs/BSCS/>

CS -   
CAS -   
OAuth -   
OOX –  
other things