WEB2.0 Accreditation and Departmental Management System

Requirements document and Execution plan

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

Ver 3.1

3/30/12

WEB2.0 Accreditation and Departmental Management System

Requirements document and Execution plan

# Table of Contents

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

2.0 Background: The ABET assessment model……………………………….3

2.1 Northern Arizona University case study………………………..4

3.0 Proposed solution…………………………………………………………………….6

4.0 Functional requirements………………………………………………………....7

5.0 Constraints and Feasibility Issues……………………………………….…..16

6.0 Project Execution Plan…………………………………………………………….17

7.0 Appendix…………………………………………..…………………………………….18

3/30/2012

# 1.0 Introduction

The Accreditation Board for Engineering and Technology (ABET1) is an accreditation organization founded in 1932 who creates standards of graduates of specific engineering and computer science degree programs of many colleges and universities in the United States. Northern Arizona University’s Computer Science program first received ABET’s accreditation in October 1996, has been continuously accredited since then. 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 deceptively simple procedure 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. Collected data must then need to be carefully archived for future retrieval. The objective of our project is to create a web-based documentation monitoring system which users ways to create, upload attachments and keep track of ABET related materials. Based on provided data, the system will auto-generate email notifications and schedules and notify users of current status.

ABET bases its accreditation approval on an extensive documentation of program’s educational effectiveness. The program’s effectiveness is quantified by utilizing various assessment tools. Some of the many possible instruments include:

* Analysis of students’ grades and performance in various courses.
* Analysis of course contents, their delivery methods, and outcome fulfillment.
* Surveys of current students, program alumni, and employees.

The complexity and extent of such task comes into play when extracting each element and considering what it is composed of. The enormous amount of data collected through these various instruments, as well as instruments themselves, is then subjected to regular meta-analysis to yield insights and improvements of program’s overall effectiveness in achieving its mission.

Managing the complex data collection and analysis processes associated with a program’s ABET plan over time is both challenging and error-prone. Breakdowns often include:

* Failure to collect certain data
* Failure to properly archive collected data
* Failure to analyze archived data on schedule
* Assigning and tracking assessment tasks

The goal of our project is to build a comprehensive system to support the following main features:

* Creation of objectives
* Creation of outcomes and assessments, and association of them with objectives
* Keeping track of current and future tasks
* Keeping track of documentations such as rubrics, surveys, and course improvement document
* Data collection and data storage
* Easy-to-understand user

# 2.0 Background: The ABET assessment model

The following figure shows a general flow of the ABET accreditation model:

Program’s Vision Statement

Objective 2

Objective 3

Objective 1

Outcome

Outcome

Measure

Data set

Instrument

Improvements

The top level of this structure is the program’s vision statement, or goals of the program promised to students. This derives objectives, outcomes, and assessment measurement methods however this statement is very vague and cannot be measured as written in most cases. In order for the program mission to be assessed, the statement is broken into smaller pieces called objectives. An objective is more detailed description of the mission statement but still represents a high-level concept. To make the concept more approachable, an objective is further divided into its subcategories or outcomes. An outcome is composed of; a detailed description of one segment of the objective, list of courses that support the description, progress assessment methods, and goals that this outcome will fulfill. Each outcome is associated with a measure that captures instruments, dataset that is used to feed instruments, and analysis methods. An instrument is a collection of dataset which will be analyzed according to various schedule, and then reflected on program improvements.

# 2.1 Northern Arizona University case study

Take NAU’s Computer Science program for example. Our vision statement states as follows:

“The Computer Science programs produce graduates who are immediately able to contribute effectively in either corporate or academic contexts. Our educational philosophy emphasizes realistic software development challenges with a focus on teaming, communication and project leadership; our curriculum promotes innovation, design, and exploration of the latest advances in the rapidly changing field of computer science.”

As explained earlier, this statement is very vague and cannot be “assessed” as written. This statement is further broken down into four objective statements:

1. Graduates are technically competent and prepared for leadership and professional practice with strength in design, problem solving, communications, and teaming.
2. Graduates are grounded in computer science and related mathematical fundamentals and prepared for advanced education and lifelong learning.
3. Graduates are have an understanding of the scope and implications of the rapid and increasing integration of software-driven technologies into personal and professional spheres of modern society.
4. Graduates integrate quickly into the workplace or advanced education due to an emphasis on high quality teaching, advising and mentoring.

These statements are more apprehensible however the scope of each statement is still large. Each objective is associated with couple of outcomes on average. For example, object 1 is associated with five outcomes:

* 1. Possess professional skills and knowledge of the software design process.
  2. Ability to function effectively in both co-located and distributed software development teams.
  3. Possess abilities to effectively communicate orally.
  4. Possess abilities to effectively communicate in writing.
  5. Abilities in creativity, critical thinking and problem identification, formulation and solving.

Each outcome has a detailed description, list of supporting courses and their contributions to the outcome, assessment methods, and assessment goals that this outcome should fulfill.

Vision Statement

Objective 1

Outcome 1.1

Dataset for Outcome 1.1

The dataset is composed of various rubrics and measuring assessment goals requires survey collections from students, program alumni, and faculties.

This model depicts just one of the five objectives we have, and each objective requires its own dataset. This yields on average, total of [1 objective x 5 outcomes x 4 assessment instruments] per objective per term.

This process is repeated every term and all of collected data need to be correctly achieved for future use. The ABET accreditation occurs every seven years. If the above represents data collection model for just one objective, then the data collected over this time period will enormous and such task is overwhelming for anyone to handle manually.

# 3.0 Proposed solution

Our approach in solving the problem is to provide users a flexible tool that can be used to create their own ABET accreditation documents and data collection methods. The system should:

* Allow users of creation of custom ABET model.
* Define/edit objectives, outcomes, measures, instruments.
* Connecting elements together into a coherent model.

This requires the system to implement two major parts: task manager and archive manager. The task manager handles:

* Auto-generation of tasks based on model.
* Ability to assign tasks to individuals.
* Monitoring task computation statistics.
* Send task reminders.
* Auto-generation of tasks based on model:

Most tasks are reoccurring at some point in future, therefore, when a task is created it needs to create copies of itself in future time frame. This needs to be done automatically so that no tasks will be overlooked.

* Ability to assign tasks to individuals:

All tasks need to be assigned to individuals, such as faculties, department deans, and administrators. Administrators therefore need ability to assign tasks to system users.

* Monitoring task computation statistics:

Data without interpretation is useless. In order to make improvements to courses and its department, data needs to yield statistical representation of how assessment measures are performing. Graphs and grade distributions are examples of how data will be interpreted to quantify given measures.

* Send task reminders:

Faculty members are swamped with different tasks during semesters and tendency to forget ABET related tasks are high. To prevent any tasks to be forgotten, the system will send task reminders automatically according to the pre-determined reminder schedule. Reminders are sent both before and after (if tasks are not fulfilled) tasks are due.

The archive manager handles the following:

* Store and organize all data collected.
* Customize form building and handles for collection.
* Visualization of archive status and completeness.
* Support for analysis and report generation.
* Store and organize all data collected:

As mentioned earlier, the ABET accreditation occur every seven years. The data collected in between accreditations, as well as previously collected data must be stored and organized in a retrieval manner.

* Customize form building and handles for collection:

Because various forms such as rubrics, surveys, and syllabuses must be attached as assessments, the system must support forms building and proper collection methods.

* Visualization of archive status and completeness:

Graphical representation of data collection status and how complete they are is a great way to get an overall picture of the progress without going into progress details. The system will incorporate a graphical time line to represent the big picture.

* Support for analysis and report generation:

Based on collected data, the system will generate its analysis and concluding reports. This presents users a better idea as to how they should make changes or improvements to courses.

# 4.0 Functional requirements

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

* Authentication
* Access Control
* Clean Display
* Create Objectives
* Modify Outcomes
* Modify Assessment
* Scheduling and Task Management
* Versioning
* Uploading Documents
* Data Collection
* Authentication

The system will provide a simple way to enforce control over resources. The login system will provide support for both Central Authentication Service (CAS)2 and Open Authorization (OAuth)3 capabilities. Third-party authentication system is also supported, so it is up to organizations to choose their authentication preference.

**Use Case 1: User login**

Scenario: Bob tries to log in.

Actor(s): Bob, an administrator of A1 University’s CS department.

User Steps and system responses:

1. Bob fires the ABET ADMS.
2. Bob is presented with either CAS or OAuth authentication screen.
3. Bob enters his user name and password.
4. The system displays the main UI.

System Behavior: The main architectural component in play here is the Authentication System of choice.

* Access Control

The system will provide access based on user roles. There are three major user categories; administrators, faculties, and guests. Administrators have the ability to activate and assign users to a program. Faculties can create or modify objects, create measures and attach them to objectives. Guests’ access level is read-only. They will not be able to create or modify any fields.

**Use Case 2: Access control**

Scenario: Bob tries to log in and activate a new user Chris.

Actor(s): Bob, an administrator of A1 University’s CS department. Chris is 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 system recognizes him as an administrator.
4. The screen displays the user interface with ‘activate user’ option.
5. Bob selects the ‘activate user’ option.
6. Chris is now activated.

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.

* Clean Display

The system will provide a web-based graphical user interface to facilitate user interaction and system management. The interface will allow users to interactively define and manage all data and tasks associated with the ABET accreditation process. The interface will provide access to varying perspectives based on usage scenarios and user privileges. The different perspectives will be:

* Class: The system will provide access to specific class information that will identify how well a class has met the requirements of its associated outcomes. This will be accomplished through access to assessment instruments.
* Course: A course perspective will be provided that identifies the intended outcomes associated with each course.
* Faculty: A faculty perspective will be provided that will allow for the identification and updating of faculty tasks and their current status for completing them.
* Administrator: An administrator perspective will be implemented that provides functionality for creating and modifying outcomes, measures, and faculty tasks.

**Use Case 3: Task management display**

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

Actor(s): Bob, an administrator 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.

* Create Objectives

Users will be able to create and modify program objectives for their respective program of study. Functionality for creating and modifying the various components of program objectives will be available to administrators. The following objective components will be available for creation and modification:

* Objective number
* Objective Title
* Objective statement
* Objective description
* Objective program identifier
* Objective outcomes (SEE BELOW)
* Modifications to objectives will be versioned if necessary (SEE VERSIONING)
* Objectives will have up to n associated outcomes

**Use Case 4: Creating objective**

Scenario: Bob wants to create department objective.

Actor(s): Bob, an administrator 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 objective’.
3. Bob enters the objective to text field.
4. Because Bob is an administrator he has an ability to either increment the major version or save it as a minor version.
5. Bob selects ‘save as a minor version’ and exits.

System Behavior: The main architectural components in play here are the User Interface (UI) module, the Objective module, and the Version module. The UI handles the front-end display including the workspace, buttons, and menus. The objective and version modules handle the creation of objective and appropriate versioning.

* Modify Outcomes

Users will be able to create and modify outcomes for their respective program of study. A given set of outcomes will be linked to a defined program objective. Functionality of creating the following components of program outcomes will be available to administrators. The following objective components will be available for creation and modification:

* Outcome number
* Outcome name/statement
* Supporting courses
* Outcome description
* Outcome assessments
* Outcome rationale
* Modifications to outcomes will be versioned if necessary (SEE VERSIONING)
* Outcomes will have up to n associated outcomes

**Use Case 5: Modify outcomes**

Scenario: Bob wants modify outcomes associated with objective.

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 ‘objective’ that is associated with the outcome that he wishes to modify.
3. Bob selects outcome and select ‘modify’.
4. Bob enters changes to the text field.
5. Because Bob is an administrator he has an ability to either increment the major version or save it as a minor version.
6. Bob selects ‘save as a minor version’ and exits.

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

* Modify Assessments

Users will be able to create and modify mechanisms for assessment for a program that they are associated with. A given set of assessments will be linked to a defined program objective. Assessments will be in the form of:

* Rubrics: Associated components for rubrics will be:  
  ---Purpose  
  ---Reviewer information  
  ---Rating system identification  
  ---Definition of objectives and criteria for meeting the elements of the scoring system  
  ---Comments
* Surveys: Associated components for surveys will be:  
  ---Description  
  ---Educational objectives  
  ---Ratings for how well objectives have been met
* Course Improvement Documents (CIDs): Associated components for CIDs will be:  
  ---Overview and course statistics  
  ---Course outcomes and assessment  
  ---Course improvement strategy
* All assessment tools can be linked to n outcomes

**Use Case 6: Modify assessments**

Scenario: Bob wants to modify assessments.

Actor(s): Bob, an administrator of A1 University’s CS department.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects ‘objective’ that is associated with the assessments that he wishes to modify.
3. Bob selects assessment and selects ‘modify’.
4. Bob enters changes to the text field.
5. Because Bob is an administrator he has an ability to either increment the major version or save it as a minor version.
6. Bob selects ‘save as a minor version’ and exits.

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

* Scheduling and Task Management

The system will provide scheduling and task management. Users will be able to login to the system to check what tasks they are responsible for as well as the status of their associated tasks. Administrators will be able to check the status of all tasks. The system will be able to remind faculty members by email of upcoming tasks that need to be completed. The system will also send out varying surveys based on a predefined schedule.

**Use Case 7: Task management**

Scenario: Bob wants to check which tasks is Chris responsible for.

Actor(s): Bob, an administrator of A1 University’s CS department. Chris is 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 ‘see all tasks’.
3. The system displays the list of all tasks and faculty members associate with them.

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

* + Versioning

The system will provide versioning. Users will be able to create versions of certain objects. The system will keep track of major and minor revisions. Major revision pertains to changes made to the objectives, outcomes, and/or assessments that affect the departmental structure. Minor revision pertains to small changes made to objectives, outcomes, and/or assessments that do not affect the departmental structure. Administrators will be able to set major revisions and faculty will be able to set minor revisions. Administrators will be able to track of revisions by certain users, programs, dates, and semesters. Version-able objects will be:

* University
* Program
* Course
* Objective
* Outcome
* Survey
* SurveyQuestion
* Question
* CourseOffering
* CID
* Minute
* Assessment
* Rubric
* RubricQuestion

**Use Case 8: Revisioning objective.**

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

Actor(s): Bob, an administrator of A1 University’s CS department.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects ‘objective’.
3. Bob selects ‘modify objective’.
4. Bob enters changes in the text filed.
5. Because Bob is an administrator he has an ability to either increment the major version or save it as a minor version.
6. Bob selects ‘save as a major version’ and exits.
7. The major version number is incremented. Their current version is now ver2.0.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the Version module. The UI handles the front-end display including the workspace, buttons, and menus. The version module is responsible for keeping track of all versions, both minor and major.

* Uploading Documents

The system will provide a mechanism for uploading documents and attach them to components of our data model. Users will be able to upload various types of documents such as pdf, doc, docx, etc. Users will be able to view all documents associated with objects that are allowed to have documents associated with them. Users will be able to upload documents associated with course offerings such as:

* Course syllabus
* Surveys

**Use Case 9: Uploading documents.**

Scenario: Bob wants to upload a pdf contains survey information to CID.

Actor(s): Bob, an administrator of A1 University’s CS department.

User Steps and system responses:

1. Bob fires up the system and enters login information.
2. Bob selects ‘see all documents’.
3. Bob selects ‘CID’.
4. Bob enters the CID field and select ‘upload file’.
5. Bob selects a pdf from his local file and selects ‘upload’.
6. PDF file is now uploaded onto CID.

System Behavior: The main architectural components in play here are the User Interface (UI) module and the CID module. The UI handles the front-end display including the workspace, buttons, and menus. The CID module is responsible for keeping track of all uploaded documents.

* Data Collection

The system will provide mechanisms for collecting data in a meaningful manner. The system will be able to generate reports and export reporting information to a usable format, such as pdf, doc, docx, etc. Users will be able to backup and restore data from the system.

**Use Case 10: Reporting.**

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

Actor(s): Bob, an administrator 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 pdf document’.
3. Bob selects data to fill the document from drop down menu.
4. Bob selects ‘export’ and a pdf containing selected data is created.

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 HTML4/CSS5/JavaScript6 must complete loading in less than 5 seconds or display a loading message. No API7 call can exceed 30 seconds in execution time. The auto-shared feature of the Google Datastore[[1]](#endnote-1) 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.

# 5.0 Constrains and feasibility issues:

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

Python7, JavaScript, HTML, and CSS will be used in this project. This application will work with the services provided by the Google App Engine7, 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 OSX7, Linux7, and Windows7, by means of the Google Python SDK7.

* 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.

# 6.0 Project Execution Plan

|  |  |  |
| --- | --- | --- |
| Milestone 1 | Requirements and Execution plan document | 2/17/12 |
| 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 |

# 7.0 APPENDIX

<http://www.abet.org>

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

1. 1ABET- The Accreditation Board for Engineering and Technology, Inc., is a non-governmental organization that accredits post-secondary education program in applied science, computing, engineering, and engineering technology.

   2CAS– Central Authentication System; is a single sign-on protocol for the web. Its purpose is to permit a user to access multiple applications while providing their credentials only once.

   3OAuth– Open Authorization; is an open standard for authorization. It allows users to share their private resources stored on one site with another site without having to hand out their credentials, typically username and password.

   4HTML- Hyper Text Markup Language is the main markup language for web pages.

   5CSS- Cascading Style Sheets is a style sheet language used to describe the presentation semantics of a document written in a markup language.

   6Java Script- is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

   7API- An application programming interface is a source code based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables.

   8Google Datastore- a data storage method provided by Google for developers to store and serve data from their Google Cloud Storage service.

   9 Python-is a general-purpose, high-level programming language whose design philosophy emphasizes code readability.

   10 Google App Engine- is a platform as a service could computing platform for developing and hosting web applications in Google-managed data centers.

   11OSX- Mac OS X is a series of Unix-based operating systems and graphical user interfaces developed, marketed, and sold by Apple Inc.

   12Linux- is a Unix-like computer operating system assembled under the model of free open source software development and distribution.

   13Windows- is a series of operating system produced by Microsoft.

   14Google Python SDK- Python platform to implement application on Google App Engine. [↑](#endnote-ref-1)