Curricular Change Management with Git and Drupal

A Tool to Support Flexible Curricular Development Workflows

Abhishek Tirkey
Computing, Informatics, & Decision Systems Engineering
Arizona State University
Tempe, AZ USA
abhishek.tirkey@asu.edu

Kevin A. Gary
Computing, Informatics, & Decision Systems Engineering
Arizona State University
Tempe, AZ USA
kgary@asu.edu

***Abstract*—The Software Enterprise at ASU aims at equipping students with practical knowledge of modern software practices through a multi-year instructional sequence that employs an iterative feedback pedagogical model in order for graduates to be adequately prepared for the workforce upon graduation. By means of the Software Enterprise Website, the community of teaching practitioners in this area having similar beliefs and approaches can find a great deal of support and free resources for the purpose of Software Engineering Education. The goal of this project is to incorporate a versioning tool to a Content Management Framework (CMF) that allows the website administrator to create, moderate and publish content revisions, particularly that of the Curricular Modules which comprise of content representing topics offered in courses, while versioning the actual content using GitHub, a web-based repository hosting service. In this case, the CMF used was Drupal 7, which is PHP-written and open-source, and provides a back-end framework for a broad range of web sites worldwide from blogs and microsites to collaborative social communities. The versioning module was developed in PHP while employing the Drupal 7 API for menu structuring and CMS integration, and the GitHub API for versioning and performing easy syncing with contents on GitHub via a RESTful interface.**

Keywords—e-learning; change management; software engineering education

I. Introduction

The modern discipline of Software Engineering comprises of computing challenges that cannot be tackled without a strong grounding in fundamental theories and acquaintance with cutting-edge tools in a rapidly evolving industry. With that in view, traditional Computer Science education often falls short of meeting these "standards" due to an outdated and imbalanced approach, placing greater emphasis on theoretical learning as opposed to the practical. To counter this problem, The Software Enterprise at Arizona State University uses a project-based, multi-year instructional sequence aimed at equipping students with hands-on experience in applying methods and tools as per modern development practices. The innovativeness of the Software Enterprise can be credited to its iterative feedback pedagogical model [1], which fosters the student's skills by systematically interweaving lectures/discussions with practical application through phases of activities that contribute to better contextual learning. The materials for each of these activities are provided in the form of content types called Curricular Modules, where each module represents a certain topic of a course.

Initially, each of the individual Curricular Modules were in the form of static web content which posed a two-fold problem: the lack of a secure and robust revisioning and versioning capability. The highly iterative nature of the Software Enterprise pedagogy requires that learning content be revised and updated from time to time. Furthermore, it necessitates the availability of a tool that enables the teaching community to evolve and contribute extensions to the the existing education content, i.e, one that provides distributed version control for the various Curricular Modules [2]. The Drupal 7 Content Management System would deal with the first problem by classifying each of the Curricular Modules as individual nodes of a general content type and then treating each section of a module as editable fields within that node. However, the latter problem would require the writing of a fresh tool in PHP to be incorporated into the Drupal CMS. This would be feasible by making use of the GitHub API, a RESTful interface that allows developers to write applications to retrieve information from GitHub repositories as well as make changes on GitHub.

II. Background

A. Content Management using Drupal

With the explosive growth of digital information over the World Wide Web and various information repositories, the collection, management and delivery of this information would be a massive challenge for most website developers if they were not being met by additional tools called Content Management Systems (CMS). Content Management comprises of a number of processes starting with the creation and editing of contents, approval of created or edited contents before publication, delivery for publication, and eventually, the association of content with metadata in order for it to be easily referenced [3].

Along with Content Management Systems, software tools that support rapid and effective development of web applications are of significant interest to developers. These tools are collectively called Web Application Frameworks, and eliminate much of the overhead of core processes involved in web development, enabling the programmer to perform faster

978-1-5090-5756-6/17/$31.00 ©2017 IEEE
SERA 2017, June 7-9, 2017, London, UK

development with improved overall application structure through making effective use of reusable components [4].

Ideally, a system that shares aspects of both Content Management Systems and Web Application Frameworks would be most suitable for this project. Fortunately, such a system can be found in Drupal, an open source Content Management Framework (CMF), that aims to be a unified system balancing the flexibility of Web Application Frameworks with that of the simplicity of most Content Management Systems. One of the essential facets of Drupal is that unlike most other Content Management Systems, it provides the platform and tools to set up customized features on the website by interfacing with a common underlying system, as opposed to isolating different website components. For instance, it allows for the creation of different content types such as news articles, blog entries, multimedia pages, etc., each of which can be customized according the user's needs, instead of using additional plugins for each, while also providing the capability to reuse or blend aspects of these features to create new components.

Drupal manages content by controlling information flow between several layers within the system. The data pool forms the base of the the system, which comprises of nodes which represents sets of related information about the data to be displayed on the web page, such as body text, title, content, author info, tags, date, etc. Most content types (blog posts, news items, static pages, etc.) are treated as variations on the concept of nodes, while the information architecture is maintained through the menu system, taxonomy and views. The data may be additionally managed or processed through the use of Modules, and the theme system, which is the topmost layer, controls and provides the external appearance of the website to the public.

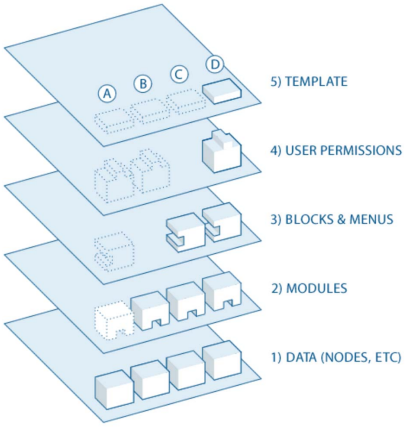

Fig. 1. The Drupal Flow (from [5])

Modules are plugins that build on Drupal's core functionality. A Drupal module comprises of a collection of functions that link into the Drupal system, thus incorporating additional functionalities to the Content Management System. In general, three kinds of modules can exist on a Drupal site - modules that ship with Drupal, modules that are contributed by members of the Drupal community, and modules that are custom written by the site developer [6]. In the case of this project, the versioning tool would be built as a Custom Drupal Module, i.e, the third category, employing the Drupal API while maintaining Drupal coding standards; this would also mean that the tool would be incorporated into the Drupal menu system in order to provide easy access to its interface.

B. *Version Control - Git and GitHub*

GitHub is a popular repository hosting service comprising of millions of source code projects, claimed to be the "largest host of source code in the world" [7]. It supports Git-based distributed version control and source code management and security features such as access control along with various means for developers to collaborate on projects. The Git version control system allows application developers and content authors to add modifications to their work while keeping revisions straight and storing changes in a central repository with the ability to view complete history and perform full version tracking, while also providing support for non-linear development via rapid branching and merging.

Although Github is primarily employed by developers for the versioning of source code, it is also applicable in a variety of other domains [8]. In this report, the service will be seen to be primarily utilized for hosting and versioning content data of the various Software Enterprise Curricular Modules.

GitHub also offers a web service that allows developers to create GitHub-based applications in the form an HTTP-based Representational State Transfer (REST) API. In a nutshell, RESTful web services provide client systems (in this case, the Drupal web tool) the ability to "access and manipulate textual representations of web resources using a uniform and predefined set of stateless operations" [9]. Requests to API access made via the GitHub web service is done over HTTPs and elicits responses in JSON format. Depending on the HTTP method and parameters passed, records on the GitHub API may or may not be changed. The GitHub REST API would be put to effective use in this project as the Custom Drupal Module invokes HTTP methods through the use of a GitHub API Client to retrieve and manipulate content data and metadata associated with specific GitHub repositories.

III. METHODOLOGY

A curricular module comprises of the information and materials offered in a single topic of a Software Engineering course, which is rendered on the Software Enterprise website in the form of HTML content. Each curricular module comprises of several content fields that contain information as required by the pedagogy such as course materials, guidelines for instructors, or project information.

A. *Identifying the problem*

First, although the Software Enterprise Website was initially set up using Drupal to offer a presentable appearance to the user, the website contents were at large, not optimally organized with respect to the modularity the CMF offers. For instance, each of the Curricular Modules were simply set up in the form of static content as nodes of a general content type, as opposed to being classified as a unique type. Second, in spite of the availability of contributed modules that provide revisioning of content via the Drupal interface, a need arises to write a custom module that leverages a popular distributed

versioning control system for instructors to contribute changes to the the respective Curricular Module remotely, and also allowing the website administrator privileges to moderate and publish those changes, apart from the ability to track the history of the state changes the content went through.

B. Evaluation of tools and aproaches

The first part of the problem could be immediately taken care of by the existing functionality provided by the Drupal CMS itself. One of the salient features of Drupal is that it allows for the creation of custom content types with the ability to add and define multiple fields for a content node. In this case, since all Curricular Modules share a common layout and content structure, the creation of a new content type with a defined set of fields pertaining to different aspects of the Curricular Modules is ideal.

However, incorporating version control for the content would be more non-trivial by comparison, as several considerations must be taken into account while designing and developing the tool. First, as per the requirements stipulated by the instructor, the tool should be integrated into the Drupal CMS itself, which would mean two things. First, it would have to be built bottom-up as a Custom Drupal Module and employ the Drupal API to implement a significant part of the interface as well as aspects of the tool's functionality (such as programatically updating nodes after content verification). Second, since Drupal along with its Modules is PHP-based, it is needless to say that most of the logic involved in the tool would have to be written in PHP as well, including the portion leveraging the GitHub REST API in order to implement content versioning on GitHub.

The Drupal 7 API is an essential component of the Content Management Framework in use as a whole and provides the building blocks to develop custom modules and subsystems, such as re-usable core module routines for the purpose of programming user interfaces, storing and retrieving data, handling events, caching, and so on. To develop a custom module, one would have to first get acquainted with the concept of hooks [10], which allows modules to extend the Drupal core, allowing the module developer to use all the functions, variables and structures in core, while also defining new hooks to be used by other modules. In more technical terms, a hook is simply a PHP function that follows a name structure that includes the module name followed by the hook separated by an underscore, with a specific return type and predefined set of parameters.

As far as the functionality dealing with content versioning via GitHub is concerned, two considerations must be taken into account - finding and employing a suitable third-party GitHub API Client library for PHP, and the appropriate format to store the content data on remote repositories for convenient access, editing, pulling and rendering on the site.

For the first part, it would be useful to glean the libraries section of the GitHub Developer Website [11] that enlists a number of third-party libraries to leverage the GitHub API for a variety of frameworks and programming languages. In this instance, for PHP, several libraries were available, the most popular one being PHP GitHub API 2.0 developed by KnpLabs [12], alongside packages for frameworks such as Joomla!, Kohana, Nette, Laravel. The key would be to select a library that made minimal use of additional packages and frameworks to avoid complications and compatibility issues with the Drupal API as well as to make the module code more maintainable and reusable for developers not well-versed with such tools.

Second, since the Curricular Module content would be stored in files on GitHub repositories, the format in which the files should be written and stored in comes into question. In order to choose the best format, we would have to take into account the following

1. Would the format be readable enough for instructors to view and perform edits?
2. Would it allow easy parsing and conversion to HTML to be rendered on a webpage? In other words, can the developer make efficient use of pre-existing libraries to render the files on a webpage as desired without going additional lengths to write extra logic in code?

Based on these two requirements, we can rule out using simple plain text formatting to represent content data as it does not meet the second one, although its usage of basic character sets allows for easy viewing and modification. The idea is to use a text notation that strikes a balance between being human-readable and machine-readable. For our scenario, the following three formats would be considered:

1. Extensible Markup Language (XML) - this is a popular markup language used across several information domains. Typically, an XML document comprises of a string of characters, which is divided into markup and content as per syntactic rules defined by XML standards. In content publishing, employing XML would mean simply "marking up textual content with semantically meaningful elements conveying the author's intent, thereby defining the document structure" [13]. However, although the design of XML is aimed towards documents, it poses readability problems for performing prompt viewing and editing due to its strong syntactic nature, which is one of the reasons why it is being more and more adopted for other purposes (such as representation of data structures, outsourcing of metadata and attribute-values, etc.)
2. JavaScript Object Notation (JSON) - Considered by many to be an effective alternative to XML, JSON represents data objects as attribute-value pairs [14] that eliminates the use of tags to mark data while having a much smaller grammar compared to XML, therefore being more human-readable. It matches the data model of most programming languages, therefore machine readability is not an issue. However, for the current application, we require the certain elements of the content data to be distinguished in order to represent HTML semantics (such as markers for tables and links for rendering purposes) and the lack of extensibility of JSON makes it inconvenient to do so.
3. Markdown - This is a lightweight markup language following a simple syntax with plain text formatting, and is

used in applications where the final rendered output needs to be viewed alongside the raw document. It is designed in a way so that it is extremely human-readable and easy to convert to HTML and other formats using a Markdown parser. It is widely employed in web-based publishing services such as weblogs, where the data is entered via a simple input interface, which is converted to HTML by the server software. For the current application, it would make good sense to employ markdown formatting in text files, then using a parser such as PHP Markdown [15] to generate HTML content from those files.

```
<curricular_module module_id="1" title = "Software Design
Overview">
    <field title="Discussion">
        <row>Software Design Overview Part 1 <![CDATA[<a
href="/sites/default/files/CST315/fall2012/notes/DesignOvervie
w1.ppt" target="_blank">PPT</a>]]></row>
        <row>Software Design Overview Part 2 <![CDATA[<a
href="/sites/default/files/CST315/fall2012/notes/DesignOvervie
w2.ppt" target="_blank" style="line-height: 1.44;
background-color: transparent;">PPT</a>]]></row>
    </subheading>
</curricular_module>
```

```
{
  "module" : "Software Design Overview",
  "field" : "Discussion",

  "paragraphs" : [
    {
      "text" : "Software Design Overview Part 1",
      "href" :
"/sites/default/files/CST315/fall2012/notes/DesignOverview1
.ppt"
    },
    {
      "text" : "Software Design Overview Part 2",
      "href" :
"/sites/default/files/CST315/fall2012/notes/DesignOverview2
.ppt",
    }
  ]
}
```

```
  * Software Design Overview Part 1.
[PPT](/sites/default/files/CST315/fall2012/notes/DesignOvervi
ew1.ppt)
  * Software Design Overview Part 2.
[PPT](/sites/default/files/CST315/fall2012/notes/DesignOvervi
ew2.ppt)
```

Fig. 2. Example showing Curricular Module content data represented in XML, JSON and Markdown respectively

C. *Proposed Solution*

In terms of the Drupal API, for this project we shall be utilizing the Menu System, Theme System and Entity API. First, the Menu System will be useful in defining the navigation menus for the tool's user interface and also for routing page requests to routines based on URLs specified in the menus via the callback system. For instance, if the administrator wishes to view a particular Curricular Module that is pending for verification, they would have to first click on the link on the Drupal Menu that brings them to the main menu of the Versioning Module, then click a link on the menu item that brings them to a page where they can view the pending modules, and so forth. All this structure is defined by the Menu System, which also specifies the functions that gather data to be displayed on the respective pages. The Theme System primarily deals with allowing the developer to use functions and templates to directly control Drupal's output, and in this case it would facilitate the writing of raw PHP code in order to flexibly provide logic to not only display required elements but invoke methods outside Drupal's API and leverage additional tools such as the GitHub API client libraries. Finally, when the Drupal nodes representing Curricular Modules needs to be updated after being verified by the administrator, the Entity API would help in simplifying the handling of nodes by providing wrapper classes to update the Curricular Module fields conveniently.

The GitHub REST API would be leveraged using the GitHub API Easy Access developed by Miloslav Hůla [16], which is a developer-friendly library in that its usage is rather intuitive for any PHP programmer and is quite undemanding in terms of getting acquainted with its workflow. It gains access to the GitHub repositories of a user via the specification of an access token, and requests are sent via HTTP methods in the form of a string containing the URL and parameters, all specified by the GitHub API documentation. Moreover, all the data is returned in raw JSON format as described in the documentation, which facilitates referencing for carrying out parsing in order to retrieve the required information to perform the respective operation.

The text data representing Curricular Module content could be stored in plain text file format, but would include Markdown notation for effective parsing and rendering to HTML when required using the PHP Markdown library. This would come into play when the administrator wishes to view commits on GitHub rendered in an identical format as displayed on the public section of website, especially while moderating the changes that were committed. For the instructor's convenience, the tool will also include a section comprising of SimpleMDE, an open-source Javascript Markdown editor that can be easily incorporated in the tool for editing changes to files in the last commit.

IV. IMPLEMENTATION

Initially, all of the website's contents, including the individual Curricular Modules had to be ported to ASU WebSpark, a new Drupal 7 environment pre-loaded all the themes and modules required for a web standards compliant, accessible and mobile ready ASU Drupal site [17]. This time, however, instead of simply posting the Curricular Module content under a general page type, a new content type named 'Curricular Modules' was created and individual fields were added to its structure that represented the various parts of the content, which would not only provide a more organized layout while performing updates on the content but also identify the individual node fields' machine name in order to programmatically alter the content through the Drupal API. (See Fig. 3 & 4.).

Software Process

Submitted by atirkey on November 2, 2016 - 11:47am

General Information:

Title	Software Process
Version Info	Version 1.0, submitted by asingh64 on 09/30/2012 at 02:28 PM
Author	Kevin Gary
Mapping to Content Taxonomy	**SEEK:** PRO **CS2013:** SE/Software Processes
Abstract	This module discusses the different software lifecycle models such as waterfall model, iterative and incremental model, spiral model, RUP and Agile/XP. It goes on to give a summary on the pros and cons of each of the models.
Size	**Preparation:** 1 hour Reading **Preparation Quiz:** 30 minutes **Discussion:** 75 minutes **Practice:** 2 hours 30 minutes
Comments	

Learning Objectives:

- Understand the phases of the Software Process.

Topics:

- Process Phases
- Process Lifecycle Models

Preparation:

- READING: Wikipedia Software development process WIKI
- READING: (Optional): The Agile Manifesto
- QUIZ:
  - IMS QTI 2.0 format ZIP
  - Moodle XML format XML
  - XHTML Format HTML

Discussion:

- Software Process Lecture Notes. PPT

Fig. 3. A Curricular Module page with content

LABEL	MACHINE NAME	FIELD TYPE
Title	title	Node module element
URL path settings	path	Path module form elements
Body	body	Long text and summary
General Information	field_cm_general_information	Long text
Guidelines for Instructors	field_cm_guidelines_instructors	Long text
Learning Objectives	field_cm_learning_objectives	Long text
Topics	field_cm_topics	Long text
Preparation	field_cm_preparation	Long text
Discussion	field_cm_discussion	Long text
Practice	field_cm_practice	Long text
Projects	field_cm_projects	Long text
Reflection	field_cm_reflection	Long text
Other Resources	field_cm_other_resources	Long text

Fig. 4. Drupal interface showing content fields with machine names

The custom module was developed using the guidelines specified in the Creating Custom Modules section of the official Drupal 7 documentation [18]. The module files would reside under the sites/all/modules/custom directory in a folder with a name that represents the module's machine name, which would be used while implementing hooks as previously described in this report. The central module file that would be used to implement the Drupal API and control execution of the module's overall functionality would be given the extension .module with the module's machine name as the filename.

The Drupal Menu System is utilized in implementing a tree menu structure for the user to browse through pages rendering the various functionalities of the tool, such as showing the modules pending for approval, viewing previous versions of modules and performing changes to content files belonging to the latest commit on GitHub, etc.

Since this module would be an important site tool, it should be easily accessible from the Drupal main menu. On logging in, the administrator is able to access the module by clicking a link with the module's name on the Drupal main menu (See Fig. 6), which directs them to a page comprising of a menu suggesting the basic uses of the module, which leads them to the respective pages to perform activities pertaining to the versioning of the Curricular Modules. For example, on clicking the 'Pending Modules' link, the user would be directed to a page comprising a list of all the Curricular Modules pending verification, which in turn are individual links to pages that display the latest version of the Module in HTML format with the options to publish or discard the change.

Currently, the functionalities provided by the module are 1) Verification of Pending Modules - to allow the administrator to view latest changes to module files on GitHub and publish these changes to the website, 2) Make changes to files on GitHub via an user-friendly interface with a Markdown Editor, and 3) Viewing all commits and branches on GitHub for every Curricular Module, in the form of generated HTML content. This section will briefly describe their workflow and implementation.

The 'Make Changes' section allows the administrator to browse the files belonging to the latest commit on GitHub in order to make changes to the files directly. When the link from the Module's main menu is clicked, the program sends an HTTP request to the GitHub API via the GET method to receive JSON data that contains the list of repositories, each representing a Curricular Module, along with related metadata, which is then parsed in order to generate a list of hyperlinks with the names of each Curricular Module on them. When one of these is clicked, a similar process is repeated, this time to render a list of the fields to be edited. On clicking the link named edit for any one of these fields, the user is directed to a page where they can make changes to the field using a Markdown Editor. The field contents would be rendered on the Editor on page load by first requesting information to receive the encoded file contents via GitHub API, saving the decoded contents in a plain text file and then reading that file to display the markdown text on the text box. On clicking save, the program uses the HTTP PUT method to perform changes on the GitHub repository.

The 'View Commits and Branches' section was similarly implemented by generating page elements from retrieved data through the use of the PHP GitHub API Client library. It follows a similar workflow as found on GitHub, except that it allows the user to view entire past Curricular Modules in the same format as that of the published content.

'The 'Pending Modules' section of the module is by far the most important functionality as it grants the administrator privileges to moderate the updates pushed on GitHub in order to be published on the site. The workflow begins with the instructor making changes to the files belonging to latest commit on GitHub, either through the use of the "Make Changes" section of the Versioning Module, remotely via Git on their local machine and then performing a push, or through the interface provided on GitHub itself. This results in a push event being triggered on GitHub, which is received by the

specific repository's Webhook. The Webhook then sends an HTTP POST payload to a PHP script on the website's server, which was written as part of the module. This script then pulls the repository files onto the site server, in a directory named 'pull_folder'. When the website administrator views the 'Pending Modules' section, a list of the Curricular Modules represented by the repositories pulled to that directory will be displayed as hyperlinks. Either of these links will lead to a page where the relevant files of the particular repository will be rendered to display the Curricular Module in proper HTML format. Upon perusing the changes done in the last commit, the administrator can then choose to either verify the changes, discard the changes or leave the decision for later, by clicking on the appropriate button. On clicking the "Yes" button, the program then employs Drupal's Entity API wrapper classes to make changes on the respective node to publish the content, and writes a tag on the current GitHub commit with the message "Synced with website contents", to mark the last point of sync with the published content. Pressing the "Discard" button deletes the pulled directory representing that Curricular Module on the server and then removes the reference to the latest commit of that repository on GitHub, thereby performing a reset to the previous commit. The page also includes a list of links to unverified commits from the point of last sync, if any, as it is possible for instructors to perform one change after another without the content data being verified and published on the website.

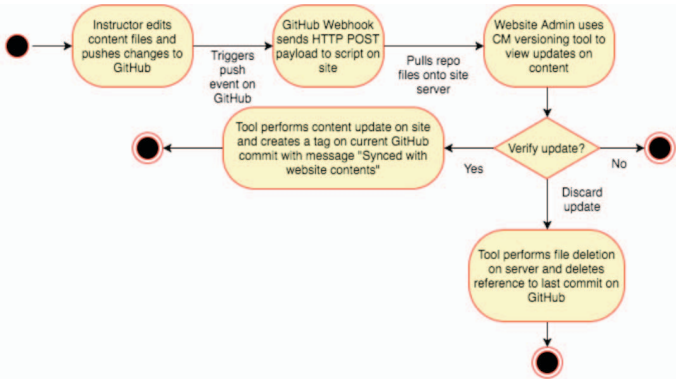

Fig. 5. Workflow of the Pending Modules section of the Versioning Tool

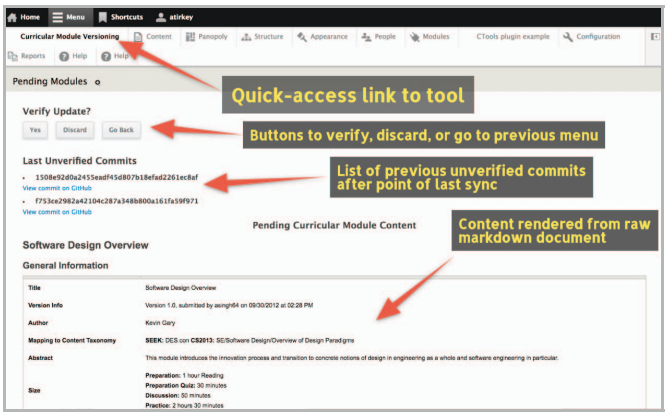

Fig. 6. The Pending Modules section of the Versioning Tool

V. VALIDATION

The tool was validated by Subject Matter Expert review (SME), in this case faculty that teach in the Software Enterprise courses.

The Versioning Tool was tested thoroughly to ascertain whether all implemented functionalities executed successfully without generating any errors or warnings and that all requirements were met. In specific terms, the following checks were performed for functionality testing:

- The updates pushed by the instructor onto GitHub is reflected in the Versioning Module, with the name of the respective Curricular Module visible on list of pending modules and the content rendered on the verification page accurately portraying the data.
- Following the verification of a pending Curricular Module, the changes are successfully published on the website in expected formatting, and its title is removed from the Pending Modules list.
- If a pending Curricular Module is discarded, the reference to the respective commit is deleted on the GitHub repository relating to that Module, and the previous commit is restored as the last visible commit.
- If a pending Curricular Module is discarded, its title longer visible in the Pending Modules list, unless the previous commit before it was not verified, in which case it is now on hold for review.
- The website administrator is able to successfully make changes to files representing Curricular Module content that belong to the most recent commit on GitHub through the 'Make Changes' functionality.
- The website administrator is able to view previous commits of individual Curricular Modules rendered in the same format as that displayed on the website's published content's page.

Since most of the development of this tool was done on a local server, rigorous testing had to be carried out after deployment to the remote server. One of the catches that surfaced during deployment was that the PHP Github API Client library required the PHP version to be at least 7.0 in order to function properly, whereas the PHP version running on the remote server was found to be 5.6.5. The problem was resolved after having performed the version upgrade on the server.

VI. CONCLUSION

Apart from the fact that the functionality provided by the Content Versioning Tool caters significantly to the iterative nature of the Software Enterprise pedagogy, several other benefits can also be taken into account. First, the availability of such a tool reduces workload of individual instructors considerably, since they will now be able to contribute extensions and changes to the learning content remotely through a distributed version control system such as Git. Moreover, the tool provides a user-friendly interface allowing the website administrator to moderate and approve changes that were pushed remotely by an instructor. It is important to

also note that since the instructors involved in the Software Enterprise pedagogy are strongly influenced by the modern Software Engineering principles of Agile development, and have applied the practices to the pedagogy itself, it can be said that the versioning tool enhances the previously incorporated Agile practice of Continuous Integration in the Software Enterprise where the iterative pedagogy "integrates the delivery of new concepts with their practice, context and reflection" [18]. Another key aspect is that the platform utilizes open-source (Drupal), leaving open possibilities that a future commercialization service model may be built around the technology [19].

REFERENCES

[1] Kevin A. Gary. The Software Enterprise: Practicing Best Practices in Software Engineering Education, *The International Journal of Engineering Education*, April 25, 2008, pp. 1-12.

[2] Kevin A. Gary, Srikesh Mandala. Distributed Version Control for Curricular Content Management, IEEE Frontiers in Education, 2013, pp. 1-3.

[3] Robert Emer Broadbent — A Functional Framework for Content Management, BYU ScholarsArchive, All Theses and Dissertations, Paper 1737, June 16, 2009, p. 11.

[4] Mathias Schwars. Design and Analysis of Web Application Frameworks, Aarhus University, PhD Dissertation, January 29, 2013, p. 3.

[5] *Drupal Overvie*w [Online] https://www.drupal.org/docs/7/understanding-drupal/overview (November 29, 2016) [accessed Dec 1, 2016]

[6] *Module developer's guide* [Online] https://www.drupal.org/developing/modules (November 20, 2004) [accessed Dec 1, 2016]

[7] Georgios Gousios, Bogdan Vasilescu, Alexander Serebrenik,, Andy Zaidman — Lean GHTorrent: GitHub data on demand, Delft University of Technology, Eindhoven University of Technology, 2014.

[8] *Not Just for Coders: 9 Ways To Use GitHub For Creative Work* [Online] http://www.makeuseof.com/tag/just-coders-9-ways-use-github-creative-work/ (March 3, 2014) [accessed Dec 1, 2016]

[9] *Representational state transfer*, Wikipedia, The Free Encyclopedia [Online] https://en.wikipedia.org/w/index.php?title=Representational_state_transfer&dir=prev&action=history (17 August 2004 — 1 December 2016) [accessed Dec 1, 2016]

[10] *Drupal API - hooks* [Online] https://api.drupal.org/api/drupal/includes%21module.inc/group/hooks/7.x (2014) [accessed Nov 30, 2016]

[11] GitHub Developer - *Libraries* [Online] https://developer.github.com/libraries/ (2016) [accessed Nov 30, 2016]

[12] *PHP GitHub API 2.0* [Online] https://github.com/KnpLabs/php-github-api (September 8, 2016) [accessed Nov 30, 2016]

[13] Philip Fennell — Extremes of XML, XML London 2013 Conference Proceedings, University College London, London, United Kingdom, 2013, pp. 80-86.

[14] *ECMA-404 The JSON Data Interchange Standard*, http://www.json.org. [accessed March 10, 2017]

[15] Michel Fortin — *PHP Markdown* [Online] https://michelf.ca/projects/php-markdown/ (March 1, 2015.) [accessed Nov 30, 2016]

[16] *Github API easy access library* [Online] https://github.com/milo/github-api (2014 — 2016) [accessed Nov 30, 2016]

[17] *Drupal @ ASU* [Online] https://drupal.asu.edu (2016) [accessed Nov 30, 2016]

[18] *Creating Custom Modules*, Drupal [Online] https://www.drupal.org/docs/7/creating-custom-module (2000-2017) [accessed March 10, 2017]

[19] K. Gary, H. Koehnemann, J. Blakely, C. Goar, H. Mann, A. Kagan. A case study: Open source community and the commercial enterprise. Proceedings of the 6th International Conference on Information Technology: Next Gerenation (ITNG'2009), pp. 940-945, Las Vegas, April 2009.

.