A Support System for Generating SCORM Compliant Open Source Software Usage Manuals

Akhmad Syaikhul Hadi¹, Takashi YUKAWA²,*, Yukikazu MURAKAMI³

¹Nagaoka University of Technology, 1603-1 Kamitomioka-machi, Nagaoka-shi, Niigata 940-2188, Japan
²Kagawa National College of Technology, Department of Electrical and Computer Engineering, 355 Chokushi-cho, Takamatsu-shi, Kagawa 761-8058, Japan

Abstract
Open Source Software (OSS) is software whose source code is open to the public. Currently, OSS is widely used in many aspects of IT society. Because OSS development is community based, unlike commercial software, the lack of good documentation or the maintenance of manuals is one of the main problems of using OSS. Due to its rapid development, OSS manuals become easily obsolete. Moreover, the installation or the usage varies depending on the operating system. To solve the documentation problems, Murakami et al. proposed a method of automatically generating a web manual for installing an OSS by editing the log information recorded during the installation process. Unfortunately, the web manual generated by this system was not suitable for wide use in learning management systems. Therefore, this paper extends the system by Murakami et al. to one with the ability to deliver an automatically generated Web manual on an e-learning management system, modify the content of the manual, and skip unnecessary information in the learning process.

Keywords: open source software, installation manual, SCORM, learning management system

1. Introduction
Open Source Software (OSS) is source code that is available to the public. Anyone can view, modify, improve, and redistribute the source code. OSS holds numerous advantages, such as low cost at deployment, high-quality software, and relatively easy customization. Therefore, OSS is widely used for purposes such as system development and academic research.

However, OSS development is community based. Even though most developers want to uphold the quality of the software, most OSS systems lack good documentation or manuals. Consequently, the quality of the documentation is often forgotten. Furthermore, with the rapid development of OSS, the documentation becomes obsolete as soon as a new version of software comes out. Consequently, the end-user has to access a forum or another end-user to obtain the information needed for troubleshooting.

E-mail address: yukawa@vos.nagaokaut.ac.jp.
To solve the problem of insufficient documentation, Murakami et al. proposed a method of automatically generating a manual for installing OSS by using the log files created during the installation process by a trained user [1, 2]. In general, an OSS installation in the Linux operating system is done by using a Linux command via a terminal or a console. The trained user issues a script command to record everything that is output to the terminal and places it in a text log file and also an OSS screen casting, Pyvnc2swf, to record screen motion and save it as a video log. By calculating each time the command is entered on the terminal, the user splits the video log into a segmented scene that is associated to the command and saves the video log in a unique directory. Then the user combines all unique directories as Web-based documentation, which is referred to as the OSS usage manual. The output is shown in Fig. 1.

![Web-based documentation example](image)

The goal of this study is to extend the system proposed by Murakami et al. to enable generated Web-based documentation to be delivered in any e-learning system, such as a learning management system (LMS), and to allow the documentation author to create its content.

The proposed system in this paper also aims to achieve flexible navigation for learning the installation manuals. Knowledge and skills on the open source software platform (Linux) differ among users. Even document-type installation manuals consist of two parts, quick installation and step-by-step (or detailed) installation. The proposed system achieves both types of documentation modes with one learning content, because the user can dynamically switch between quick navigation and step-by-step navigation. For example, a user who has sufficient knowledge for Linux file operation, but lacks knowledge of SQL, can learn with the quick mode for most steps, then switch to the step-by-step mode for database operations.

2. Overview of SCORM

A simple way to deliver learning content into any e-learning system is by implementing the e-learning standard. Among the existing standards for learning contents is the Shareable Content Object Reference Model (SCORM) standard. The SCORM standard was developed by the Advanced Distributed Learning Network (ADL). The SCORM standard specifies a framework for content that has durability, portability, reusability of Web-based learning content, plug-and-play interoperability, and accessibility.
SCORM is not really a standard by itself; rather, it is a collection of the specifications and standards produced by the Aviation Industry CBT Committee (AICC), IMS Global Learning Consortium, and the Institute of Electrical and Electronics Engineers (IEEE). SCORM implements the Learning Object Metadata (LOM) from IEEE standards and the content packaging specifications from IMS. It also defines a Web-based learning Content Aggregation Model (CAM) and a Runtime Environment (RTE) for learning objects. CAM provides a way to identify and describe the learning content by defining the following components: asset, sharable content object (SCO), and aggregation. An asset is the electronic representation of media, text, images, sounds, web pages, or other pieces of data that can be delivered to a Web client. An SCO, which represents a collection of one or more assets that are the smallest logical unit of information delivered to learners via the LMS, can also communicate with the LMS. An aggregation is a collection of related activities that may contain SCOs or other aggregation. The organization is the part of the content package where SCOs are ordered into a tree structure and assigned sequencing behaviors.

All SCORM content is placed in a ZIP file or Package Interchange File (PIF). As shown in Fig. 2, a SCORM content package consists of two principal parts.

1. The XML manifest file that describes the following:
   - All of the SCOs or assets included in the package
   - A representation of the content structure (called the organization)
   - The sequencing rules
   - Metadata for the SCOs, aggregations, and the package itself

2. All of the actual SCO and asset files for the content package.

Several LMSs compatible with SCORM 2004, which is the newest version of the SCORM standard, have been developed and used in a variety of educational institutions. These LMSs include ILIAS, ELECOA, Sakai, and OpensourceLMS. ILIAS replaces OpensourceLMS as the official LMS at the Nagaoka University of Technology. ELECOA is used at the Open University of Japan. Sakai, which is a very popular LMS, is also compatible with SCORM 2004 and is used in numerous institutions.

3. Proposed System

3.1. Required functions for the system

In general, the flow of the proposed system is illustrated in Fig. 3. The system has two main functions: content editing and SCORM conversion. The content editing function assists the manual author in modifying and updating the contents. This function also enables the flow of the content organized by skipping unnecessary or
detailed information for the quick mode. The content editing function is visible because it is the user interface (UI) between the manual author and the proposed system. The SCORM conversion function converts an OSS usage manual into a SCORM compliant manual.

The system is designed as a Web-based application, so it can be accessed through any modern browser. The system provides an upload page to send the OSS usage manual as input. If no problems arise, the system proceeds to the editing page that enables the manual author to edit and update the content, skip unnecessary or detailed information, and generate the SCORM compliant manual.

3.2. Implementing the content editing function

The OSS usage manual has a good directory structure for organizing its contents. The navigation of all commands is stored in one file. In addition, the content associated with each command is stored in a unique directory and its name has a prefix step, and a number starting from 1, such as step1, step2, etc. By using regular expressions, the system retrieves the contents for editing and updating the contents to the server via AJAX. Also, it is possible to skip any unnecessary content (command) that is not needed in the learning progress. The UI of the authoring feature is shown in Fig. 4.
3.3. Implementing the SCORM conversion function

One of the objectives of SCORM is to create reusable content. The contents of the OSS usage manual are structured in a hierarchical fashion. When converting the OSS usage manual to a SCORM compliant manual, the implementation should maintain the existing structure of its content with little modification. As shown in Fig. 5, the content of each command is placed in a specific directory with the prefix name `step`. Therefore, the content of each command can be interpreted by the SCO directory name because the command forms a granular object that can be tracked and delivered by the LMS.

![Fig. 5. The directory structure of the OSS usage manual](image)

To generate the manifest file, the version of the SCORM standard should be declared first in a `metadata` element. In this paper, the latest SCORM standard, SCORM 2004 3rd Edition, is used. Unlike the previous version, SCORM 2004 has navigation and sequencing features. Then, the `organization` consists of multiple activities (SCOs or aggregation) represented by the `item` element. Each `item` element `title` represents the terminal command name on the list iteratively, and the `identifierref` attribute refers to the SCO listed in the `resources` section of the manifest file. Inside the `organization`, the sequencing control mode is set to `Choice=true` (allows learners to select the order in which they view the content), `Flow=true` (requires learners to view the content in an order defined by the documentation creator) and `ChoiceExit=true` (controls whether a learner can choose an activity outside the active aggregation). That is the basic manifest implementation.

The implementation of skipping unnecessary or detailed information inside the OSS usage manual is done by using the sequencing and navigation features in SCORM 2004. Sequencing is the process in which content objects (SCOs) are selected by the LMS for delivery to the learner. During the sequencing process, rules are evaluated and an activity (SCO) is identified for delivery. In general, sequencing is initiated by a navigation request. The LMS provides a navigation element in its own UI and the user can switch between skipping navigation (quick mode) and non-skipping navigation (step-by-step mode) at any time. For a logical way to skip features, the first step is to set up a global threshold value via a dummy SCO, which sets the visibility to false. This dummy SCO is then put at the first node under the `organization`. Then, for every unnecessary or detailed command (SCO), the skip rule should be set up and described in the manifest file. This skip rule is used when a measured threshold value is less than the global threshold. In this case, the system does not load the command, skips the current activity (SCO), and moves to the next activity (SCO).
As shown in Fig. 6, after conversion to a SCORM compliant form, the OSS usage manual can be delivered in any LMS that is SCORM 2004 compliant.

4. Evaluation

To evaluate the proposed system, a task assignment followed by a questionnaire survey was conducted. The examinees for the evaluation were six people. The objectives of the evaluation were to compare the easiness level in editing the contents between the examinee’s favorite text editor and the proposed system, and to identify the usability of the design of the user interfaces. The evaluation also observed the effect of skipping unnecessary information in terms of understanding and speed of the learning process.

As the first task assignment, the examinees were instructed to insert an explanation for each step of the OSS usage manual. Each examinee used their favorite text editor for this activity. Specifically, each examinee created a text file and wrote a description of the operation with his/her favorite text editor for each of the installation steps. The same activity was next done in the proposed system. According to the comparative result shown in Fig. 7(a), two of the six users judged the proposed system to be easier than using their favorite text editor. In Fig. 7(b), 67% gave the response that the design of the user interfaces was easy to use.
As the second assignment, the examinees learned the SCORM compliant OSS usage manual. The SCORM compliant OSS usage manual itself contains unnecessary and necessary information in equal proportions, such as mistaken command inputs to distract the examinees from achieving the learning objective. The learning process was divided into a normal sequence and a skipping-enabled sequence. In the normal sequence, the examinees learned all the contents, and in the skipping-enabled sequence, all unnecessary contents were hidden and did not appear in the learning process. The result is shown in Fig. 8. After omitting the unnecessary information, as shown in Fig. 8(a), most of the examinees appeared to understand the contents more easily. In Fig. 8(b), all the examinees learned the content faster after the skipping-enabled sequence as judged by the task completion speed.

![Fig. 8](image)

**Fig. 8.** (a) Comparison of the understanding level of the skip-enabled content and the normal content; (b) The learning speed

### 5. Conclusion

This paper showed how to generate an OSS usage manual that is SCORM compliant due to its SCORM compliant status. The proposed system was able to deliver the OSS usage manual in various e-learning platforms without much effort. Through the proposed system, the manual author can create an OSS usage manual for beginners and advanced learners by skipping unnecessary information. By exploring the SCORM standard, it is still possible to develop a more interactive OSS usage manual by integrating SCORM data model.

### References