Continuous Integration in the Development and Operations of a Learning Management System

Ryota Fukuda[1,a)] Shoji Kajita[2,b)]

Abstract: It is always troublesome for both developers and operators at Application Service Providers (ASPs) to upgrade their running application services. In order to handle the upgrades with only limited human resource at the central ICT organization of Japanese higher educational institution, we propose the use of the continuous integration method to both the development and operations to automate the complicated upgrade procedure. To show the effectiveness, we have developed a continuous integration environment for our Sakai-based learning management system, using Jenkins, a continuous integration server, and Chef, an infrastructure automation system. This paper describes our on-going practice on continuous integration from the standpoint of a small team.

Keywords: e-Learning, Learning Management System, DevOps, Continuous Integration, Open Source, Sakai

1. Introduction

The use of ICT at higher educational institutions is becoming inevitable recently in any aspects of activities for teaching, learning and research. This trend makes application services at higher educational institutions more complex and required to provide more business centric applications to support teaching, learning and research, rather than just communication infrastructures and tools like campus-wide network and e-mail system. Recently, the most widely used application services for teaching and learning at higher educational institutions is Learning Management Systems (LMSs) and its adoption rate in the United Status is over 90%[1] and the one in Japan is over 50% according to our recent survey for Academic Cloud Computing in Japan[2].

At Kyoto University, since April 2013, we have started the use of *Sakai Collaboration and Learning Environment (CLE)*[3], which is simply referred to as "Sakai", to provide a Web-based LMS for students and faculty. Our Sakai is named as "PandA (People and Academy)" by students awarded for a total design competition of Sakai including logo, naming and its concept (Fig. 1). The introduction of PandA was the retirement of WebCT Campus Edition 8, which is another proprietary LMS used in the world and supported by SCSK Corporation in Japan. The transition from proprietary based LMS to Open Source based one also indicates the necessity to change our development and operation strategies on LMS, because Sakai is a complex Open Source system written in Java and has over 0.8 million LoC (Lines of Code) divided into 50–100 pluggable modules, each providing a tool of use for higher education, such as syllabi, assessments, grade books and so on. The Sakai Community, mainly based on the institutions and individuals from the member institutions of the Apereo Foundation[4][*1], actively develops it.

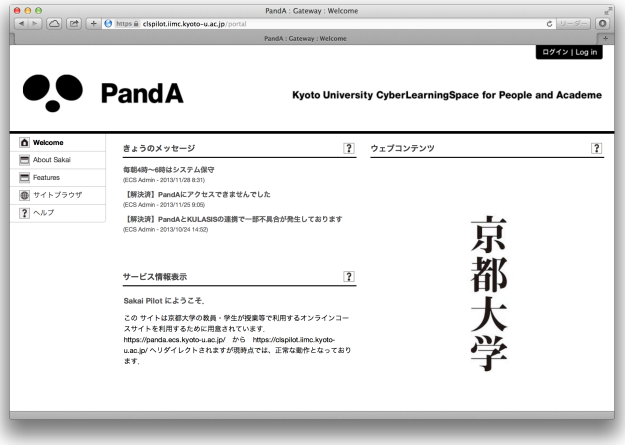

Fig. 1 PandA as the Learning Management System at Kyoto University.

Despite the complexity of the software and the number of its users in the university, we have only few engineers in charge of the development and the operations of the LMS. They have to develop tools customized for the use in the university. In special, the adoption of Open Source software has been increasing in order to reduce ICT costs and

1 Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University
2 IT Planning Office, Institute for Information Management and Communication, Kyoto University
a) rfukuda@kuis.kyoto-u.ac.jp
b) kajita@media.kyoto-u.ac.jp

*1 An umbrella organization for Open Source projects for higher education. Currently, seven Open Source projects are actively working in Apereo Foundation.

more importantly have the control of the cost, specifications and the future by higher educational institutions[5] so that we need to find a concrete way that is usable widely in the community to tackle issues posed by limited resources.

In this paper, we describe our challenges for the use of Open Source in a scarce resource envrionment and propose the use of the continuous integration method to both the development and operations to automate the complicated upgrade procedure. To show the effectiveness, we have developed a continuous integration environment for our Sakai-based learning management system, using Jenkins, a continuous integration server, and Chef, an infrastructure automation system. This paper describes our on-going practice on continuous integration from the standpoint of a small team.

2. The Problems and Our Approach

2.1 Probrems posed by current workflow

Usually, in order to get up and running a LMS like Sakai CLE at any higher educational institutions, we have three kinds of tasks for engineers in our section: development, testing (quality assurance), and operations. These steps of tasks are usually sequenced in this order and iterated several times (Fig. 3):

Development Development is the first step that the engineers write codes for the software to implement new features and/or to fix problems. The most tidious works at this stage is to incorporate two directional modifications (patches) at source codes: one is the results in the Sakai Community and the other is the ones at Kyoto University (See Fig. 2). The Sakai Community annually release the latest version and maintain two released versions. In other words, we need to upgrade Sakai every two years at least. The longer period we have for development phase, the more patches we need to manage in the two directions (See Fig. 2).

Testing (Quality Assualance) Testing is the step to ensure that the features are properly implemented and the problems are correctly fixed in the development step. If a problem is found in this phase, the process returns to the development step and the engineers fix the problem.

Operations Finally, in the operation step engineers deploy the software and serve it to the users. If a problem is discovered in the operation, it is also fixed in the development step.

Through the development, testing and operations phases, some issue tracker is usually used to manage the processes. The Sakai Community and Kyoto University are using Jira. However, the issue tracker is just tracking the status of issues and the actual works in the three phases must be done by engineers. When the human resources are scarce, engineers must have all of tasks in the three phases and it makes each phase longer. As the result, it costs more and more to manage a lot of patches. Furthermore, when a problem is found in the testing step, the longer the release cycle is, the more difficult it is to find the change in the code base which introduced the problem.

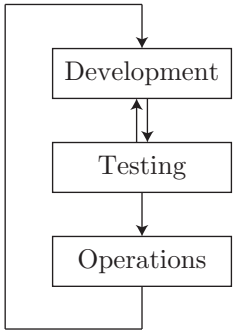

Fig. 3 Steps of workflow.

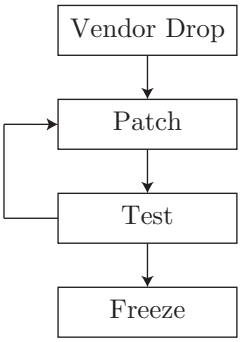

Fig. 4 Development workflow.

2.2 Our Approach

In the real world of software development, the continous integration has been widely used to treat these issues[6]. However, it highly depends on the development and operation evnronment so that each software or application service must have its own continous integration environment even if in the same organizatin.

In this paper, we propose an implementation of continuous integration for Sakai CLE used for teaching and learning at Kyoto University.

3. Proposed Implementation of CI Environment

The overview of our implemented CI environment is depicted in Fig. 5. Its important components are:

- the CI server,
- the source code repository,
- and the virtual machines (VMs).

3.1 The CI Server

The CI server controls the entire system, triggered by the submissions of patches to the repository. When a developer submits a patch, it is applied to the source tree and then built and tested in virtual environments.

There are several software solutions available for continuous integration and deployment. *Jenkins*[7] we have chosen is an open-source continuous integration server. It is actively developed by the community and has many plug-

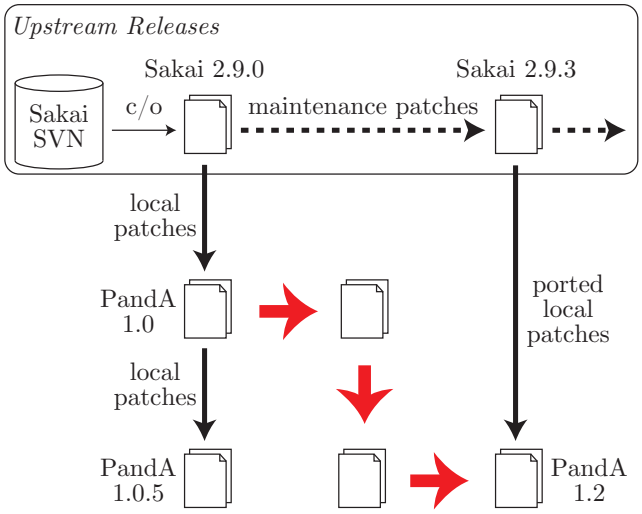

Fig. 2 Two directions to catch up the latest developments in the Sakai Community and Kyoto University.

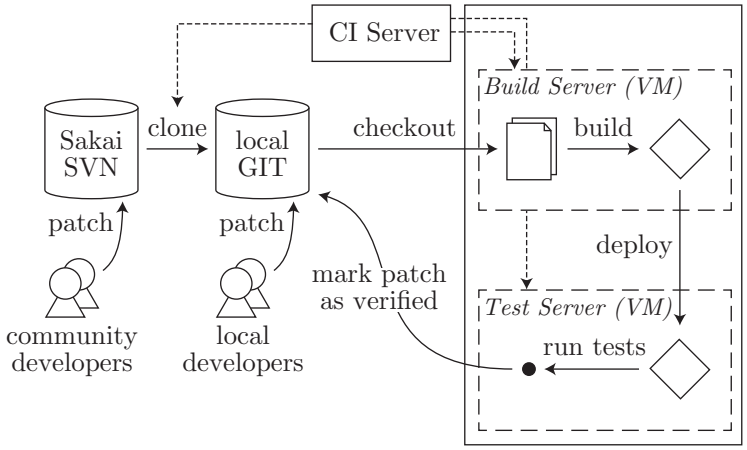

Fig. 5 Our CI environment.

ins to support various development tools, including version control systems and build automation tools and so on.

Travis CI[8] is another open-source continuous integration service. Unlike Jenkins, Travis is a hosted (or "cloud-based") CI system and integrated with GitHub, a specialized social networking service for source code management. Travis is a rather simple system but suffice for automation of building and testing of open-source products.

For our experiment, however, we want to automate the deployment of the product onto our local servers as well as automate building and testing. Thus, on-premise CI systems, such as Jenkins, are more suitable for us than hosted CI services.

3.2 The Source Code Repository

The source code repository manages patches submitted by the developers. The Sakai Community uses Subversion as the version control system (VCS) to manage their source codes. Since Subversion has centralized server-client architecture, we have to store any modifications on the existing source codes in the repository owned by the Sakai Community in Apereo Foundation. This may, however, requires legally appropriate actions regarding to copyrights and intellectual properties by submitting Contributor License Agreement. Also the centralized repository makes it difficult to store security-sensitive data, such as server configurations, in the VCS repository.

In order to avoid these troubles we now have our own Subversion repository in the local environment and uses the "vendor branch" technique that we import the original source tree in the local repository and version control the modifications on the tree (See Fig. 4). But with this technique, local revisions are disassociated with the upstream repository and it becomes difficult to automatically merge the local and upstream changes. We can overcome this technical limitation by using some distributed version control system (DVCS), with which several repositories can share revision history. So we choose *Git* as the version control system, which is the most popularly used amongst many existing DVCSes. We can use *git-svn*, a plugin to the Git,

to clone changes from the upstream Subversion repository to the local Git repository.

More precisely, we use *gerrit* as the local Git server. gerrit is a open-source code review system integrated with the Git version control system. Every change pushed to the gerrit repository is treated as a patch submission and queued for code review. After other developers review the patch and give the okay, the change can be merged to the source tree. Combining with Jenkins, we can run automated tests for every single change and verify it before it gets merged.

3.3 The VMs

We use two VMs to build and test an instance of Sakai CLE (See Fig. 5). *Vagrant*[9] is used for the creation of VMs. In order to automate the processes for building and testing, we use *Chef*[10], an infrastructure automation system. These automation are crucial for the proposed approach because the number of required steps are quite large when we use Open Source software. For example, getting up and running Kuali, which is another Open Source project for administrative systems in higher educational institutions, requires about 7,200 steps[11].

3.4 Improved workflow with CI introduced

The essence of continuous integration is automation. Automating the development routine reduces the cost of iterations of development cycles and it makes us possible to release software products in shorter spans. Unfortunately, the Sakai code base does not have sufficient automated tests. There are too few unit tests to cover the entire code base and no integration tests are available.

However, we need to automate the quality assurance procedure to reduce costs for the development. In the current implementation, we deploy nightly builds so that we can try the latest code base and perform manual testing, instead of introducing automated integration tests using testing automation frameworks, such as Selenium. In the PandA development process, several technical staff has been conducting QA activities along with the Jira tickets so that such automation for QA is crucial in a small team.

4. Summary and Future Work

In this paper we examined problems in the current workflows for the development and operations of an LMS in Kyoto University and discussed how we can improve the workflows by adopting continuous integration.

In the experiment, we developed a CI environment for our own use, but the development scheme for other institutions, which use Sakai, must not so different from ours. Hence, it is useful for Sakai institutions if we can automate the procedure to set up a CI environment, using the infrastructure automation systems introduced in this paper, in order to utilize our proposed system for their development. It is also effective for further evaluation of the proposition.

Furthermore, the automation system for deployment that we developed runs not only in the CI environment. Since it runs in VMs, which are automatically configured, the users just have to prepare VM hosting system. So this can be also utilized in development phase to check the behavior of the software and may lower the barrier to contributing Sakai Project.

Acknowledgments

This work is partially supported by Grant-in-Aid for Scientific Research (B) (Principal Investigator: Shoji Kajita, No. 22300288), Scientific Research (A) (Principal Investigator: Tsuneo Yamada (Professor, Open University of Japan), No. 23240110), Scientific Research (A) (Principal Investigator: Toru Iiyoshi (Professor, Kyoto University), No. 25242017), and Scientific Research (B) (Principal Investigator: Syuntaro Chida, No. 90464213).

References

[1] Kenneth C. Green, Campus Computing 2007, The 18th National Survey of Computing and Information Technology in American Higher Education.

[2] "Community based Approach for Building Academic Clouds as Next-Generation ICT Environments among Universities of Japan", http://www.icer.kyushu-u.ac.jp/ac

[3] Sakai Project, http://www.sakaiproject.org

[4] Apereo Foundation, http://apereo.org/

[5] Paul Courant, "Software and Collaboration in Higher Education: A Study of Open Source Software", http://www.campussource.de/opensource/docs/OOSS_Report.pdf

[6] Alan W. Brown, "Enterprise Software Delivery: Bringing Agility and Efficiency to the Global Software Supply Chain", Addison-Wesley Professional, July 2012.

[7] Jenkins CI, http://jenkins-ci.org

[8] Travis CI, https://travis-ci.org

[9] Vagrant, http://www.vagrantup.com

[10] Chef, http://www.getchef.com/chef/

[11] Lance Speelmon, "Kuali in the Cloud", http://www.youtube.com/watch?v=s-XO1Wq4MTc