An Experience Sharing on e-Learning Platform Upgrade

Janny C.C. Ng, Sze-Wing Leung(✉), Dickson T.S. Chan, Hades C.F. Tam, Benz C.L. Sze, and Ray K.C. Wong

School of Professional and Continuing Education,
The University of Hong Kong, Hong Kong SAR, China
{janny.ng, swleung}@hkuspace.hku.hk

Abstract. Since the introduction of learning management system (LMS), the development of online learning has been changing rapidly in the past decades. With the advancement of information technology, numerous LMSs, such as WebCT, Blackboard and Moodle, had been developed. New features and advanced versions of LMSs are frequently released each year. HKU School of Professional and Continuing Education (HKU SPACE) adopted the first LMS, namely "SOUL", in order to enhance teaching and learning effectiveness since 1999. In view of enhancing the needs of a sophisticated online learning environment, SOUL was revamped based on Moodle, an open source system, and a new version named SOUL 2.0 was launched in 2011. After running for a few years, a system upgrade was taken place in 2014 to cope with the latest Moodle version 2.7, which mobile format is supported and improvements were made. With various customized features implemented to the SOUL 2.0, a set of coding standards was introduced by the development team to standardize the customization of source codes. This paper shares the barriers that the team faced during the upgrading process. It also illustrates how the coding standard and effective workflow were implemented at development phase, and the way to assure system coding and source data are migrated successfully.

Keywords: e-learning · Learning management system · Platform upgrade · Moodle

1 Introduction

HKU SPACE is committed to provide high quality educational opportunities for the communities since its establishment. Starting from 1999, the School adopted the advance from technologies to provide better learning and teaching experience for teachers and students by developing their in-house built learning management system, SOUL. It was a Web 1.0 platform which was mainly for students to download material and submit assignment files. The School started using an open source learning management system, Moodle version 2.1 to take the advantage of its Web 2.0 features of being a pedagogical driven, cost-effective and community supported learning management system (LMS) solution [9] since 2011.

J. Lam et al. (Eds.): ICTE 2015, CCIS 559, pp. 233–250, 2015.
DOI: 10.1007/978-3-662-48978-9_22

With the revolution bloom on Web 2.0 in recent decades, a variety of e-learning platforms have been introduced. In order to keep in pace with the technology and adapt different learning styles, each of those platforms could release several software updates and require frequent upgrading. While an e-learning platform always involves various stakeholders, previous studies have argued that the entire campus could have to suffer a disaster if an update procedure is not implemented with a thoughtful and thorough action plan [15].

According to Moodle.org, two versions would be published per year where major and minor updates would be applied to its LMS for adopting the updated technologies such as mobile-friendly features, improving the system performance, fixing security issues, etc. To take the advantages of the enhancements and new functions in latest versions, upgrading e-learning platform from older to newer become the only option for the higher educational institutes and is a challenge to the higher educational organization undergoing the adoption of the upgraded LMS [6].

This paper introduces a case study of the upgrade processes of our e-learning platform that comprise of complicated and huge amount of customization tasks and enhancements. It also discusses the challenges, which include schedule management, staff turnover, user involvement, launching preparations, etc., during the upgrade process. Last but not least, this paper shares good practices for developing and maintaining e-learning platforms. It is hoped that the case study and the experiences shared by the team could act as a reference for any party who prepares to update an e-learning platform, especially for major or critical updates.

2 Business Needs and Benefits on Platform Upgrade

Nah & Delgado [5] emphasized that 'Business Plan and Vision' and 'Top Management Support and Championship' are critical for a project while Petherbridge and Chapman [6] mentioned that planning on upgrading a LMS version to another requires sufficient resources. With the support from the School, the LMS upgrade project team kick-off the LMS upgrade by using the latest stable release of Moodle where many security issues were fixed, new functions were developed, performance were greatly improved and Moodle Mobile app was introduced after version 2.4. To upgrade a system, it means adding new features or installing more recent software patches to the existing LMS. It is definitely having many advantages to have a regularly upgrading on the existing LMS. For instance, users are able to get access to new features, the latest versions of software often has performance and functions improvement. Modern web access features like "drag and drop", refreshing part of the web page using AJAX and jQuery support, revamping the assignment module and logging methodology, supporting group assignment, adding book module into the core module, etc., are deployed on Moodle in the recent releases. These features also enhance user experience, in other words, improve the learning experience. In addition, we can save cost by updating system regularly as outdated system may no longer be supported by the LMS provider on bugs and security fixings.

Upgrades often include bug fixes and security patches. Therefore, staying on the upgrade path makes future upgrades easier. This reason for upgrading sounds not so

compelling but the truth is, if we are sticking with a particular system, it's best to stay relatively current. Otherwise, a future upgrade can be nearly impossible or ridiculously expensive. Why? It is because most of the providers tend to stop providing support for older versions when they launch new ones. They only write scripts that let administrators upgrade system from the previous one or two versions. If it is over six or tens versions behind, there is no simple way to jump versions because the database or programming structure may have big changes. Upgrading LMS is the least we can do to keep pace and be proactive.

3 Planning for the LMS Upgrade in HKU SPACE

Open source system does not fit with standard models of software development [8]. With the nature of open source system, like Moodle, it provides a high flexibility for higher education institutions to customize the functions easily to meet the operational needs from their teachers and learners. We had hundreds of functions customized in most of the modules in Moodle version 2.1 platform in order to satisfy the needs of our stakeholders including students, teachers and programme staff. Due to the limited resources and tight project schedule for launching the SOUL 2.0 in 2011, there was lack of documentation on the code changes, hard to trace the coding due to the different coding style used by the developers and the high staff turnover rate in team. It becomes a big challenge for the project team to upgrade SOUL 2.0 to the latest version Moodle version 2.7.

Having a good upgrade plan and upgrade procedures are keys for the success of the project. To prepare a good upgrade plan for the system upgrade project, we have firstly studied the release notes to understand the changes in Moodle before we started planning the project and estimating the resource. Secondly, we had to design the development environment such as the hardware, operation system upgrade and other peripheral software upgrade. Then, we had to list all functions that we customized and estimate the resource for revamping the customized functions with updated methodology and merge them to the upgrade platform.

In addition to the estimation for the development work, a well-designed workflow for function testing and a clear procedure are also important for the upgrade process. Although Moodle has provided some scripts for migrating data from one release to later releases, we had made customizations not only on the functions but also on the database, data migration to the latest releases becomes another big issue. A well designed upgrade procedures and enough time for trial run exercise are musts for ensuring the data integrality after the platform is upgraded. During the trial run exercise, we identified different problems on data migration and developed customized scripts to enhance the efficiency and accuracy of upgrade process, such as altering fields in database, and updating the data content to be compatible with the updated platform.

When all work are ready, some efforts are needed before the upgraded system goes live, which is to backup data file, source code and database, and it can provide a fast fallback procedure when data is corrupted during migrating to production. At the project investigation and planning stage, the latest stable Moodle version was version

2.6, the resource estimation and design were based on this version. After migrating over 90 % of the customized functions from Moodle version 2.1 to version 2.6, the Moodle version 2.7.1 stable version was released. With the use of project management system, version control system and coding standard, we could easily further migrate the customized functions to Moodle version 2.7.1. Besides, from the experiences gained from trial run exercises and the well tested upgrade procedures, we migrated our customized LMS from version 2.1 to version 2.7 within the estimated duration and the upgraded system was successfully launched. In the following sections, we are going to share experiences on maintaining and upgrading a Moodle platform.

4 Good Practices Used for Maintaining and Upgrading a Moodle Platform

The development team of this project consists of one project manager, one main software engineer, three part-time software engineers and several part-time helpers who were employed for function testing. Within the limited time frame and hundreds of customized functions, we need a very good organization of the project development. One of the most important decisions to be made by the project manager is how to properly staff the project, her major goal is to maximize value creation for a given investment [1]. How to prioritize the function migration is the most important part. Use of project management system and introducing coding standard to ensure the functions handled by different developers reach the same standard, and also can be easily followed by different developers. Moreover, how to ensure the functions and data are migrated correctly, as well as how to get user involved during the upgrade process are another important elements. These elements will be discussed in this section.

4.1 Effectively Use of Project Management System for Recording and Tracking Progress of Tasks

Project management system is defined as a change management system and a collection of documented procedures that records how the deliverables and documentation of project were approved, changed and controlled. Project management system is also defined as a series of actions added to the process of getting tasks done on projects by working with project team members to achieve the project schedule, technical performance objectives and goals [3]. To achieve the project goals and objectives, we need to develop a project plan first. By complying with the project plan, we can identify the tasks and achieve the goals easily. Project management also includes managing the implementation of the definition, project planning, implementation, evaluation and maintenance.

In our LMS upgrade project, we adopted a web-based software project management tool, Trac. Trac is an issue tracking system for software development projects. It provides an interface to subversion, an integrated Wiki and convenient reporting facilities [12]. Trac allows wiki markup in issue descriptions and committed messages, creates links and seamless references between bugs, tasks, change sets, files and wiki

pages. Also, the timeline showing all current and past project events in order makes the acquisition of an overview of the project and tracking progress very easy [2]; whilst the roadmap shows the road ahead, lists the upcoming milestones. Trac helps the project manager to keep close monitoring on each task and helps the developers to record the progress of each task in order to work collaboratively among developers as well as student helpers. It also allows the project manager to monitor the progress of the project via roadmap function in the Trac system. Figure 1 shows the roadmap for SOUL 2.0 project.

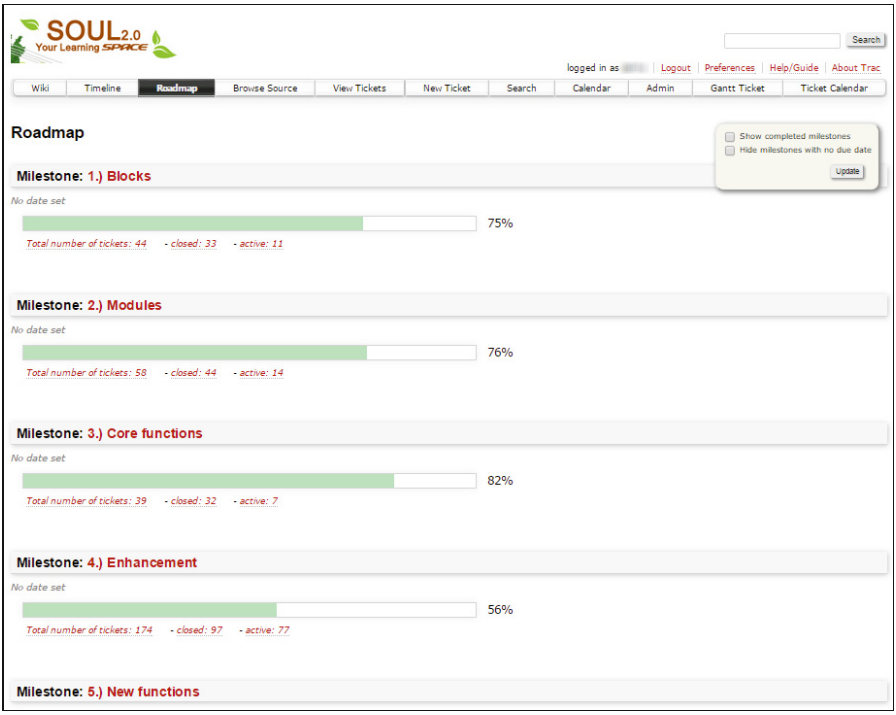

Fig. 1. The roadmap for SOUL 2.0 project

4.2 Improve Working Efficiency by Adopting a Version Control System

Crashing on coding development often happens in a collaboration project which has more than one developer. To avoid it, the project team used a software called Git to keep the version control of the source codes. Git is a distributed version control system, it supports distributed and non-linear workflows. Working directory of Git recorded complete history and full version tracking capabilities [14]. For example, there are a few developers who produce and develop the source code, they may change, extend, undo changes, and jump back to an older version, and they may need to modify the same files, the Git will keep track of the files and keep the history as version to indicate who made the change of the files.

A strict branching model designed around the project release is defined as the Git workflow. It does not add any new concepts and commands for the workflow, and it uses a central repository as the communication hub for all developers and engineers. In Git workflow, it is using historical branches structure of the project. Instead of the single master branch, it uses two branches, the master branch and the develop branch, to record the history or version of the project. The master branch saves or records the official release history while the develop branch serves as an integration branch for features.

Common conventions of Git workflow:

$$\text{Branch off (development)} => \text{Merge into : masterbranch} => \text{Release branch}$$

For example, at the beginning, there is a local repository as a clone of the Moodle 2.1 (from Moodle official site) repository and it may call the master branch, there are many other branches of the bug fix on the master branch. Figure 2 shows the relationship between master branch and other branches of the bug fix.

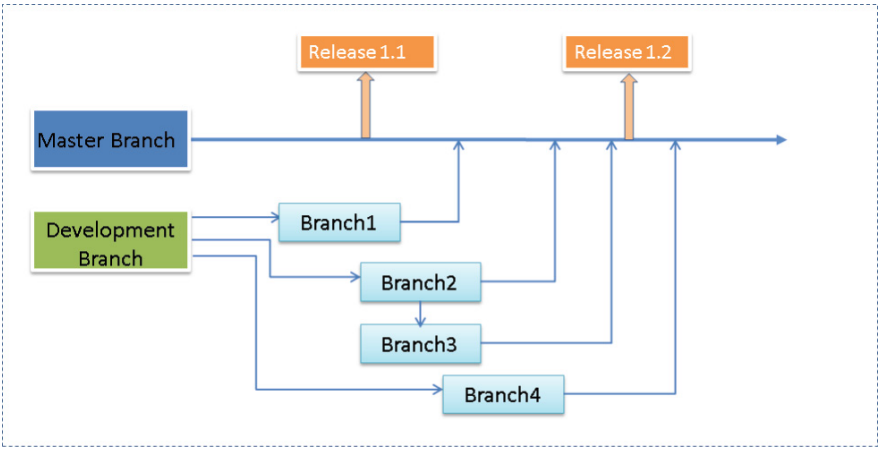

Fig. 2. The relationship between master branch, development branch and other branches

Figure 2 illustrates a simple relationship between branches. From the diagram, we know that Branch1, Branch2 and Branch4 are using the development branch. The Branch3 is basing from Branch2 because it requires some modifications from Branch2. And, once the Branch1, Branch2, Branch3 are completed and merged to master branch, a new release, named Release1.2, is launched to the production system. After that, another modification by Branch4 is performed and merged to the master branch. With using the version control system, we can easily identify the conflicts among the branches and/or revert the merged branch(es) easily with several commands. It greatly improves the working efficiency of the team.

4.3 Use of Coding Standard Making Codes Handy and Traceable

In an open source collaborative project, we would face the problems such as the turnover of technical staff, tracking the previous issues long ago and different programmers may have different coding styles, etc. Therefore, the upgrade project team setup a coding standard for the developers to follow in order to prevent the problems. Veranga [13] pointed out that implementing a coding standard on a project can make the source code more comprehensive and easy for maintaining. Also, it can allow other developers to trace the code easily. Below are the examples:

A. **Modify a function or variable inside a class.** For any code modification happens inside a class, the following steps should apply:
   (1) Change the original class name with prefix "cold_",
   (2) Create a new class to extend the class in step 1 just below the old class,
   (3) Override any variables or methods in the new class.

Original code

```
<?php
  class USER {
        function get_user() {
              return true;
        }
  }
```

Modification code

```
<?php
 //John Doe: SOUL2_00123 (23-Dec-2014) – echo message before return @Start@
  class cold_USER {
      function get_user() {
         return true;
         }
    }
  class USER extends cold_USER {
      function get_user() {
         echo "Hello World"
         return true;
         }
    }
   //John Doe: SOUL2_00123 (23-Dec-2014) – echo message before return @End@
```

B. **Modify a function not inside a class.** For any function modifications not inside a class, the following steps will apply:
   (1) Rename original function with prefix "fold_",
   (2) Create a new function using the original function name with prefix of four spaces just below the old function.

Original code

```
<?php
function get_user() {
        return true;
}
```

Modification code

```
<?php
function fold_get_user() {
        return true;
}
//John Doe: SOUL2_00123 (23-Dec-2014) – echo message before return @Start@
function get_user() {
        echo "Hello World";
        return true;
}
//John Doe: SOUL2_00123 (23-Dec-2014) – echo message before return @End@
```

C. **Other modification handling.** For any modifications not inside a class or function, an inline modification will be used with comments.
Original code

```
<?php
  echo "Helo Wold";
```

Modification code

```
<?php
echo "Hello World"; //John Doe: SOUL2_00123 (23Dec-2014) – corrected typo words
```

4.4 Prioritize on the Function Migration to Fulfill the Needs of Majority of Users

There are over hundreds of customized features built in the system in the older version. With the lack of a well documentation on the customized modules and the staff turnover problem, it is nearly impossible to migrate all customization made in version 2.1 to the newer version within a few months. Prioritizing the customized functions migration by according to its importance therefore becomes an important task. It can allow the project team to see clearly the most important task to be handled first, and which can be on hold [4]. Besides the customizations made on the core libraries that we should migrate first, the project team also decided that all customizations made on the highly used functions would be migrated to the upgraded platform. To help

understanding on the function usages among the active courses, a course function usage statistic was generated. Table 1 shows that the most popular function is uploading file resources to the course for student to download, followed by posting announcement for the latest news of the course. There were several modules not being used by any course, namely Lesson, Mindmap, Nanogong and Survey. These functions would not be enabled in the upgraded system at the beginning of the system launch until they are being well tested.

Table 1. Statistic on course usage of different module as at April 2013

Module name	No. of courses used
File resource	2175
Announcement	1697
Event calendar	1000
Folder resource	775
Label	567
Assignment	544
Turnitin assignment	490
URL	420
Grade book	203
Forum	89
Page	73
SCORM	57
Quiz	52
SWF	28
Questionnaire	24
VCLink	19
Choice	15
Glossary	14
Wiki	13
Workshop	12
Database	7
Group choice	3
Chat room	1
Lesson	0
Mindmap	0
Nanogong	0
Survey	0

4.5 A Clear Workflow for Development Deployment Can Improve Working Efficiency

A clear workflow for the development deployment can make the project run more effectively and efficiently. It helps complete the project in a timely manner, more

consistently, safely and reliably. With adopting the Trac system, the project manager would issue tickets in the Trac system and assign them to developers. The developers would investigate and estimate the resource/time required for handling the assigned tasks. After the project manager accepts the estimated resources, the deadline for the development of those tasks would be set. The developers would first pull data from development repository to their local development environment to make sure the latest coding is being used. After handling the tasks and having well tested in local environment, the developers would push to development server. The code would be reviewed by a senior developer before merging to master branch, and then the functions were tested by the helpers and reviewed by the project manager. The branches would be deployed to a staging server periodically. The staging server is an environment which simulated the production environment. All branches would be deployed to the

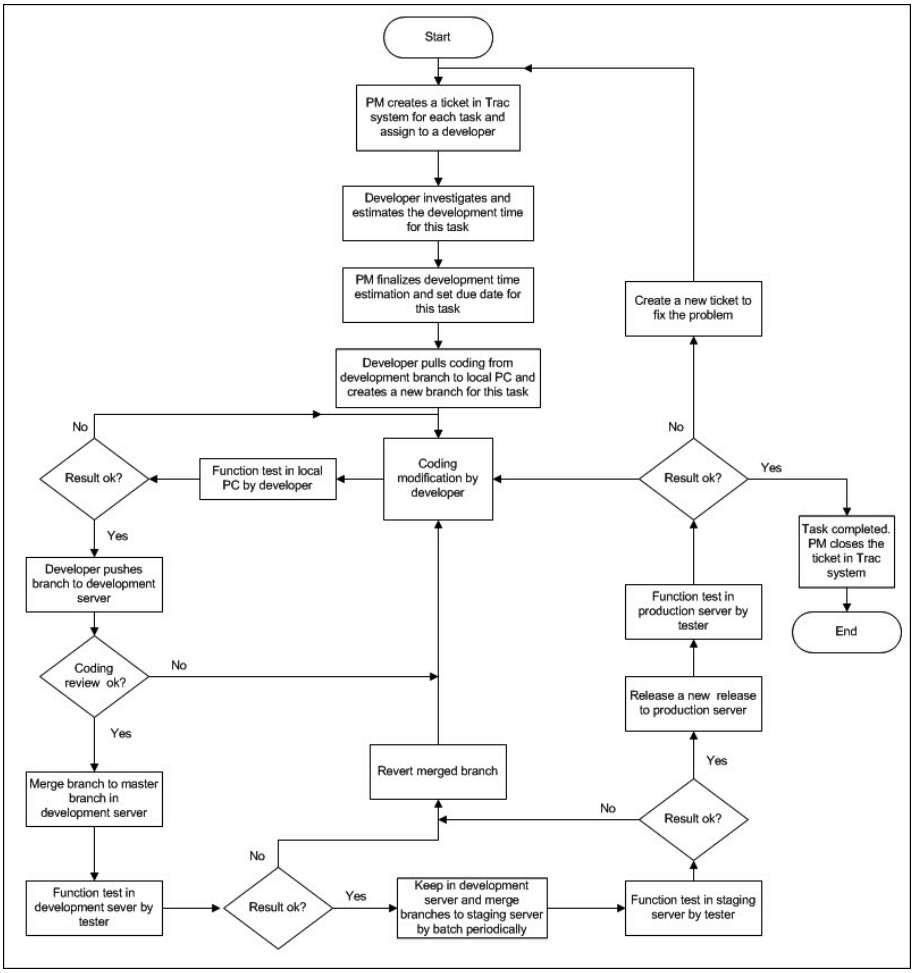

Fig. 3. Development deployment workflow

staging site for final confirmation. If there is any problem found in staging server, the merged braches would be reverted and the tickets would be assigned back to the developers for further investigation and fixing. These procedures are not only used during the system upgrade but also adopted in current system maintaining stage. The only difference is that, after the testing has successfully passed in staging server, it would launch to production site in a release. And, a final testing would be conducted in the production site. Figure 3 shows the workflow of the development deployment process.

4.6 Migration Planning to Reduce the Potential Risks of Error During Upgrade

A successful upgrade project requires a migration plan. The migration plan addresses issues associated with phasing out legacy systems and moving to the new system. These issues include user interface compatibility, database compatibility, transition support, system interface compatibility, and training. Also it involves tradeoffs between cost, schedule, risk, and resources. The migration plan should identify prototyping needs in system upgrade and which data are included in the migration process. Prototypes can effectively test the potential solution, especially in cases where current systems are complex and involve many users. The migration plan should identify prototyping needs. At the same time, it should address the extent to which migration considerations call for prototyping both to mitigate risks and to demonstrate proof-of-concept to users. A prototype can be completed in weeks as opposed to

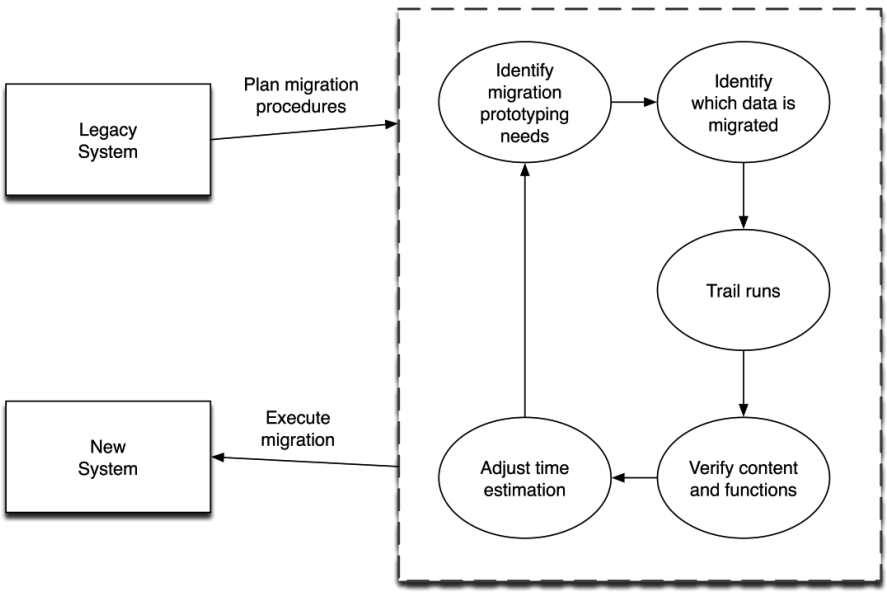

Fig. 4. Iterative migration cycle for system upgrade

months of laborious specification. An effective prototype can also be used for collecting comments from users on the new user interface and operational usage scenarios before system implementation decisions are made. In addition to identify the prototyping needs on system upgrade, it is also important to understand what types of data is migrated from the old system to the new one. Much of the data from the old server can be migrated automatically using the update tools but some are not. For those data are not convertible other tools or routine needs in order to achieve migration goals. Some of them must even be migrated manually. This may consume more time to complete task. Figure 4 shows the iterative migration cycle for upgrading a system.

4.7 Trial Run Migration Procedure to Reduce Risk and Simulate the Real Situation

In a LMS, it stores lots of information and materials in complex format. Before migration process begins, it is suggested to perform a pre-migration impact assessment to verify the cost and likely outcome of the migration. Throughout the pre-migration impact assessment, the procedures and potential risks can be identified. Besides, after several trial runs on the migration procedure, it can provide far more accurate analysis of resource requirements, the expected outage duration and help to refine the migration procedures.

During the trial run exercises, we identified over 20 issues of data and function migration problems. One of the critical issues was the server hardware requirement for performing the data migration in our upgrade process. It takes a long time to migrate the data, which have 10,000 courses and around 7,000 assignment submissions, with using a Virtual Machine server with 16G RAM and 8 vCPU. The whole process was improved by using a dedicated high performance server. Besides the hardware issue, some of the problems were due to the change of data fields in database during the function customizations in Moodle version 2.1. There were also some problems which were not handled by the migration tools provided by Moodle especially the assignment module and gradebook. Additional scripts were prepared for fixing those issues.

4.8 Time Estimation for the Upgrade Process to Ensure Accurate Estimations and Monitor Any Possible Failure at Early Stage

The amount of time taking for performing the migration or upgrade process varies based on how much data the LMS contains and how much customization LMS was configured in the old system. It helps identify the potential risks on the whole process of migration. Moreover, it allows us to obtain a more accurate time spent on each task especially for the major tasks such as installation, configurations of LMS and its plugins, customization of the new system, data migration, the upgrade process and any manual configuration tasks, testing, etc.

It is important to identify the most time consuming tasks, which makes us easier to determine whether the migration process is success or failure. For example, in trail run, migration of student assignments takes around four hours (80 % of total). However, in

Table 2. Upgraded procedures and the time estimation for each task

Task	Description	Estimated time completion		
		Hr	Min	Sec
Export database	Export database contexts from Legacy system	0	10	5
Compress database	Compress exported database	0	6	46
Transfer database	SCP compressed database to staging server	0	2	17
Decompress database	Decompress database and prepare for import	0	3	0
Import database	Import database to staging server	0	27	21
Fix data integrity problem	o Fix assignments data migration problem	0	0	20
	o Fix various database problems	0	0	9
	o Drop unused tables	0	0	2
	o Fix link problem on existing quiz	0	0	2
Git checkout to branch in stage 1	**Upgrade platform to MOODLE 2.2.11+**	**0**	**3**	**30**
Remove activities	Remove mindmap activity	0	1	30
Remove blocks	Remove customized blocks: SOUL2 block, cohort_control block, dndupload block, support_tools block, system reminder block	0	3	0
Git checkout to branch in stage 2	**Upgrade platform to MOODLE 2.6.4+**	**0**	**33**	**30**
Remove local plugin	Remove custlib, course_maintenance	0	1	40
Remove reports	Remove large scale log and report (email log, user tracking report)	0	1	30
Remove export method	Remove csv from grade export method	0	0	32
Backup database	**Backup the upgraded database as a staging backup**	**0**	**2**	**20**
Upgrade assignments	Upgrade all assignment (2.2) instance to assign instance	4	5	39
Restore database	Restore tables for customized plugins and blocks	0	2	0
Checkout DB_Stage3	**Upgrade platform to MOODLE 2.7.1**	**0**	**4**	**35**
Post upgrade DB fix	Add missing capabilities, load faq content, SCORM settings, convert course format	0	2	0
Final configuration	Enable CAS, edit CAS strings, mod enabler	0	3	0
Backup database	Backup database for production database server	0	2	20
	Total:	5	57	8

the migration it takes more than five hours, this gives administrator a signal that the migration process may be failure. Table 2 is an example of the time estimation process.

4.9 Use of Checklist for Data Migration Verification and Functional Test to Ensure Data and System Integrity During System Upgrade

A checklist with test cases was designed for verifying the data and functions migration. The checklist aimed to evaluate whether the expected outcomes have been achieved. During the trial run exercise, the data migrated was sampled checked and functions in the upgraded platform were tested by the development team. Outstanding issues were listed and fixings were applied to the upgrade scripts. Fine tune on the migration procedures and checklist were also performed during each round of trial run exercise. A sample of checklist used in the current task is shown in Fig. 5.

				Assignment										File				
Checked by	Test date	Course ID	Course code	Count in source	Count in upgraded platform	Checked URL	Submission type	Description	Allow access period	Due date	submission inbox	Download all submission	Other remarks (if any)	Count in source	Count in upgraded platform	Checked URL	Description	Other remarks (if any)
example	5/30/14	101	SO 01-101-00 (31)	3	3	(randomly pick one assignment among the 3)	same	same	same	same	same	same	NA					
Tester1	8/23/14	38064	CC 88-768-12-01 (31)	4	4	http://10.21.x.xx/upgrade/mod/assign/view.php?id=355402	same	same	NA (not set)	NA (not set)	same	same	NA	47	47	http://10.21.x.xx/upgrade/mod/resource/view.php?id=24387	same	NA
Tester1	8/23/14	41143	SE 01-118-04 (32)	0	0	NA	NA	NA	NA	NA	NA	NA	NA	0	0	NA	NA	NA
Tester1	8/23/14	42380	IT 06-133-12 (31)	2	2	http://10.21.x.xx/upgrade/mod/assign/view.php?id=359355	same	same	same	same	same	same	NA	14	14	http://10.21.x.xx/upgrade/mod/resource/view.php?id=359355	same	NA
Tester1	8/23/14	41743	SE 35-805-00 (41)	1	1	http://10.21.x.xx/upgrade/mod/assign/	same	same	same	same	same	same	NA	5	5	http://10.21.x.xx/upgrade/mod/resource/view.php?id=339716	same	NA

Fig. 5. An extract of a checklist for data migration verification and functional test

4.10 Get User Involved and Conduct User Training to Maximize User Capability and Experience on the New Platform

User involvement is one of the key factors of success of a computer system. Tait and Vessey [10] found that system complexity and resource constraints have strong effects on system success, either directly or indirectly through their influences on user involvement. Rossum [7] also pointed out that it is the best practice for coordinating the users during the migration process. A testing platform was setup for letting our users involve in verifying migrated data and testing the functions in the upgraded platform. The testing site was built with the final tested functions and the data was merged from the existing production platform. After the functional test and data verification were performed by the development team in the testing platform, users were invited to verify their course content in the testing site as well as try out the functions before the system launch.

A survey was conducted after the user verified the migrated data and functions used in their courses, 27 items related to data migration were asked for their checking and confirmation. We received the response on 21 courses, the result is presented in Table 3.

Table 3. Verify data after data migration in upgraded platform

Question		Yes (%)
I. Course materials		
1.	All Announcements are migrated correctly	87.50 %
2.	All File resources are migrated correctly	86.67 %
3.	All Folder resources are migrated correctly	93.33 %
4.	All Label resources are migrated correctly	93.33 %
5.	All Page resources are migrated correctly	90.00 %
6.	All URL resources are migrated correctly	87.50 %
7.	All VClink resources are migrated correctly	87.50 %
II. Course Activities		
8.	All Assignment and Turnitin Assignment activities are migrated correctly	90.91 %
9.	All Chat activities are migrated correctly	83.33 %
10.	All Choice activities are migrated correctly	85.71 %
11.	All Group Choice activities are migrated correctly	83.33 %
12.	All Database activities are migrated correctly	80.00 %
13.	All Forum activities are migrated correctly	80.00 %
14.	All Glossary activities are migrated correctly	80.00 %
15.	All Offline grade item activities are migrated correctly	83.33 %
16.	All Lesson activities migrated correctly	66.67 %
17.	All Mindmap activities are migrated correctly	80.00 %
18.	All Questionnaire activities are migrated correctly	83.33 %
19.	All Quiz activities are migrated correctly	83.33 %
20.	All SCORM package activities are migrated correctly	80.00 %
21.	All Wiki activities are migrated correctly	66.67 %
22.	All Workshop activities are migrated correctly	60.00 %
III. Others		
23.	All graded items are migrated correctly	80.00 %
24.	All Blocks are migrated correctly	83.33 %
25.	All course layout are migrated correctly	85.71 %
26.	All participants are migrated correctly	85.71 %
27.	All Calendar events are migrated correctly	85.71 %

From the result above, most of the data migrations of the modules had been confirmed with over 80 percent correctly of them migrated to the new platform, except Lesson, Wiki and Workshop activities. We found that user answered "No" in the survey due to the fact that the course enrollment to the user was missing in the testing site. Since no customizations were done on the Lesson, Wiki and Workshop and the utilization of the functions were very low, we assumed the migrations were correct as well.

We also asked the users to try the new platform in the testing site, feedback was collected afterwards. In Table 4 below, it was found that we received positive feedback on the upgrade platform, functions and new interface. The server performance improvement was not recognizable because of the scale of the testing site.

Table 4. Comments on functions in SOUL 2.0 upgraded platform

New functions/enhancements		Agree (%)
(a) Enhancement on Assignment and Turnitin Assignment function		
1.	Allowing user to change from assignment submission type is useful	88.46 %
2.	Group assignment function is useful	78.85 %
3.	Improvement on the file upload by using drag and drop file uploading interface is useful	86.27 %
4.	The interface of Turnitin Assignment is improved	72.73 %
(b) Enhancement on interface		
5.	The interface is clear while the "Turn Editing On" function is on	88.46 %
6.	Allow user to edit the resources/activities name in the course main page is useful	84.91 %
7.	Combining the pull-down menu for Add a Resource and Add an Activity into one pull-down menu is a good improvement	77.36 %
8.	It is a good improvement and tidy interface by collapsing most of the common settings in setting page.	69.23 %
(c) Other enhancements		
9.	More help information on different functions is provided	72.00 %
10.	New types of access restriction settings is useful	66.67 %
11.	The server performance is improved	55.10 %
(d) New functions		
12.	I want to learn more about Rubric grading	50.00 %
13.	I want to learn more about Book recourse	50.00 %
14.	I want to learn more about Feedback activity	50.00 %
15.	I want to learn more about Badges function	38.30 %
16.	I want to learn more about Copy block function	46.81 %

From the survey results, we observed that users were not eager to learn the new functions. We discussed and decided to focus on the enhancement of the existing functions.

Torkzadeh & Van Dyke [11] examined the influence of training programs on Internet self-efficacy and user attitudes toward computers and the results suggested that training significantly influences Internet self-efficacy for individuals with 'high' or 'low' attitude towards computers. The team arranged a series of hands-on training workshops before and after the system was launched. A total 153 participants took part in the workshops. The workshops were recorded and uploaded to the platform for users to review it at any time. Although the user interface of the upgraded platform is similar to the original one, we believe sufficient training would help users adapt to the upgraded platform more quickly. Ongoing training workshop will be arranged to help new users to get familiar with the system, training on basic usage and specific functions will be arranged separately.

5 Conclusion

Implementation of a new learning management system is a rather easy task when compared with upgrading an existing one. However, upgrading a learning management system is an inevitable task in education institutes nowadays to strengthen their teaching and learning support services. After a system has run for several years, performing functions customization in the LMS and technical staff turnover are unavoidable. These make it hard for the new developers to maintain and support, or even upgrade the platform. In this paper, we shared good practices for maintaining the existing platform and highlighted the important strategies on implementing the system upgrade. This includes the use of project management system to keep tracing the task development history, adopting a version control system for both collaborating development work and storing versions, use of coding standard for improving not only the quality of source code but also make the code more traceable, and defining a clear development procedure. The preparation of migration plan, trial run of the migration procedures, getting user involvement and providing users training before and after a system go lives are also the key factors of success of a system upgrade project.

Our experience on SOUL 2.0 suggested that, implementing the project management system, version control system and the coding standard would not only help us monitor the changes on the functions easily, but also provide an easy way for system upgrade. This can be evidenced by our experience on further upgrade of the system from version 2.6 to version 2.7. We would conclude that the whole process of system upgrade was shortened by 70 % with the suggested measures and upgrading the system to the latest version should no longer be a panic task for the project team.

References

1. Boehm, B.W., Sullivan, K.J.: Software economics: a roadmap. In: Proceedings of the Conference on The future of Software Engineering, pp. 319–343. ACM, May 2000
2. Brookins, M.: Benefits of Using Project Management Software (2015). Accessed from http://smallbusiness.chron.com/benefits-using-project-management-software-2196.html
3. Ghioca, T.: Advantages and Benefits of Project Management Software, 8 February 2011. Accessed from http://www.rationalplan.com/projectmanagementblog/advantages-and-benefits-of-project-management-software/
4. Gosenheimer, C.: Project Prioritization. Office of Quality Improvement, University of Wisconsin (2012)
5. Nah, F.H., Delgado, S.: Critical success factors for enterprise resource planning implementation and upgrade. J. Comput. Inf. Syst. **46**(5), 99 (2006)
6. Petherbridge, D., Chapman, D.: Upgrading or replacing your learning management system: implications for student support. Online J. Distance Learn. Adm. **10**(1) (2007)
7. Rossum, P.: Best Practices in Data Migration. Research and Services The Data Warehousing Institute (2006)
8. Sandred, J.: Managing Open Source Projects: A Wiley Tech Brief, vol. 18. Wiley, New York (2002)

9. Sarker, M.O.F., Matthews, J., Gramp, J.: On optimal strategies for the development and operation of Moodle in Higher Education Institutions. In: Moodle Research Conference, pp. 5–13 (2013)
10. Tait, P., Vessey, I.: The effect of user involvement on system success: a contingency approach. MIS Q. **12**, 91–108 (1988)
11. Torkzadeh, G., Van Dyke, T.P.: Effects of training on Internet self-efficacy and computer user attitudes. Comput. Hum. Behav. **18**(5), 479–494 (2002)
12. Trac: Welcome to the Trac Project, 4 March 2015. Accessed from http://trac.edgewall.org/
13. Veranga, L.: 10 Benefits of Using Coding Standards to Software Development Team, 22 January 2008. Accessed from http://www.articlesbase.com/programming-articles/10-benefits-of-using-coding-standards-to-software-development-team-312610.html
14. Why Git? (2015). Accessed from http://www.git-tower.com/learn/git/ebook/command-line/appendix/why-git
15. Xu, H.: Learning management system transformation. In: MWAIS 2014 Proceedings. Paper 11 (2014)