Beyond LMS: Expanding Course Experience with Content Collaboration and Smart Assignment Feedback

doi:10.3991/ijet.v5i4.1321

I. Bosnić, M. Orlić and M. Žagar
University of Zagreb, Zagreb, Croatia

***Abstract*—Tools for content collaboration among students and teachers and automated assignment verification have been introduced to students of a third-year computing course. Collaborative work on lecture slides promotes content openness to students via a new open perspective and improves the quality of lectures among other benefits. Assignment verification helps students resolve most common problems giving immediate feedback on their submissions. WikiPres and ORVViS applications developed to assist in these tasks are introduced and presented.**

***Index Terms*—e-learning, content collaboration, assignment verification**

I. INTRODUCTION

This paper discusses our experiences with tools for content collaboration and automated assignment verification developed and introduced to our students in a course on *Open computing*. The breadth of the course topics and emphasis on giving the students an overview of a wide range of concepts and related technologies made it difficult to keep track of all recent technological developments in the field. To address numerous student comments and give them the opportunity to take part in the course creation, *WikiPres* application was developed. The main goal of the application is to enable collaboration on lecture notes in a *Wiki* environment, originally created as presentations.

In the practical part of the course, students are given assignments to work on from home, and to try out various concepts introduced in the lectures. These assignments require students to use numerous technologies in a limited time, which results in different problems – from environment setup, configuration issues to common problems associated with learning a new language (programming language like *PHP*, or descriptive language like *XML*). A typical student response to these problems is time-consuming trial and error with numerous questions posted on course forums, often repeating from year to year. To help students overcome most common problems, automated verification of student assignments with *ORVViS* was introduced.

The paper is organized as follows: the *Open computing* course and the base activities in LMS are described in section 2. The efforts in making the content available for collaboration, with the *WikiPres* application are described in section 3. Section 4 gives the overview of automated verification of course assignments with *ORVViS*. Current issues observed in initial usage are given in section 5.

II. COURSE OVERVIEW

The *Open computing* course is given to ~70 third-year students of computing at our University. The students are introduced to the concepts of open systems, open technologies and their importance, as well as the nature of open culture and open licences. Some basics in open system security are also explained. The course gives a brief overview of openness in hardware, software and user experience, with an emphasis on standards, their purpose and means of establishing them in the world of distributed and interactive information services.

On the practical side, students can experience how various open technologies comprising the modern information services (web was taken as an example of user-oriented client-server computing) fit together technically over protocols (Internet and web protocols such as *HTTP*), and languages (*HTML*, *PHP*, *XML*, *JavaScript*, *Java*). Having a course focusing on openness forces us to illustrate the described concepts on our own example –lecture notes are prepared using an open source licensed *OpenOffice.org* suite, course content is published under the *Creative Commons* licence, and course assignments are carried out using open technologies and programming languages.

The course is divided into three cycles, each lasting 4-5 weeks. The first cycle deals with basic terms of open systems and client-side technologies. The second cycle is oriented towards server-side technologies, while the third is focused on topics of integration, user interfaces, dynamic aspects, security and open licences. Each cycle consists of:

- weekly lectures,
- bi-weekly course assignments (laboratory exercises),
- mid-term written exam after each course cycle.

These course elements are carried out in the classroom, as the course is designed like a mixed-mode e-learning course. However, these elements are interconnected to e-learning activities, which can be divided in two groups: core CMS/LMS elements and extended activities, which are the focus of this article. The following section shortly describes the core usage of CMS and LMS in our course.

A. *Elementary LMS experience*

For base course services, the Faculty's Content Management System – FER *E-Campus* is used, as in all of the courses at the Faculty. The news, educational resources and forums for students' questions on assignments are published there. CMS is integrated with *Moodle* LMS on both data and authentication level [1]. The usage of two systems, CMS and LMS, is due to the requirements of uniformity for all the courses, as different courses at the faculty have different levels of technology usage in education. Courses which use less e-learning concepts, publish just the content and course information through the CMS and do not use LMS. Courses with higher level of e-learning implementation increase the use of online activities with the students. Different tools, such as RSS feeds, enable us to reduce the possible ambiguity of systems, e.g. course news or repository updates from CMS are mirrored in LMS.

In this course, LMS is mainly used for the following tasks:

- embedding the lectures published on the *SlideShare* service, as a means to open our educational content to the public;
- delivering the course assignments to the students and collecting student submissions, in order to check on the submissions before the actual face to face meeting with the students.

These systems lack the features of content collaboration and assignment verification, which we find important for a better course quality. The following chapters describe our motivation for improving the course with these elements, along with the overview of applications which implement these activities.

III. CONTENT COLLABORATION

A significant part of course efforts is dedicated to making content both open and available to collaboration. The need for open content is a logical extension to the general course topic. Course lectures are made publicly available under the *Creative Commons* licence, which initiated the Faculty decision to recommend such a licence for educational content at the whole institution. With this approach, we try to come closer to the complete Open Educational Resources model [2]. We have also started to publish the course lectures to *SlideShare* to make them widely available in an online, easily-embeddable form.

In addition to teaching staff creating the content, our long-term effort is to change the students' role in the course: instead of being passive consumers of presented content, we see them as co-creators and co-editors of course content. This can be accomplished by offering the students to work, together with the teaching staff, on current lecture slides.

A. *Collaborative work on lecture slides*

The possibility of collaborative work on lecture slides brings the following benefits:

- Promote the openness on the content level, as one of the indirect course goals.
- Improve the lectures' quality by collaboratively correcting mistakes.
- Added content to the teacher's version of the slides, such as related links to tutorials, examples or different explanations. Additional content related to the slides, relevant to a student, can also be helpful to his colleagues dealing with the same topic issues.
- Enable discussion on lecture content outside lecture-time constraints. As the time for the face to face lectures is rather limited, collaborative work on slides helps students to get introduced to the topics and solve initial problems before the lectures. Also, it can foster further discussions, not only through the course forums, but directly related to the specific slides of the content.
- Keep content up-to-date, which is often a problem with evolving modern technologies. Content needs to be changed from year to year, due to the significant changes in the field being taught. Students who have practical experience with technologies used in this course can correct mistakes, give practical advices, or help other students or staff in analyzing these topics.

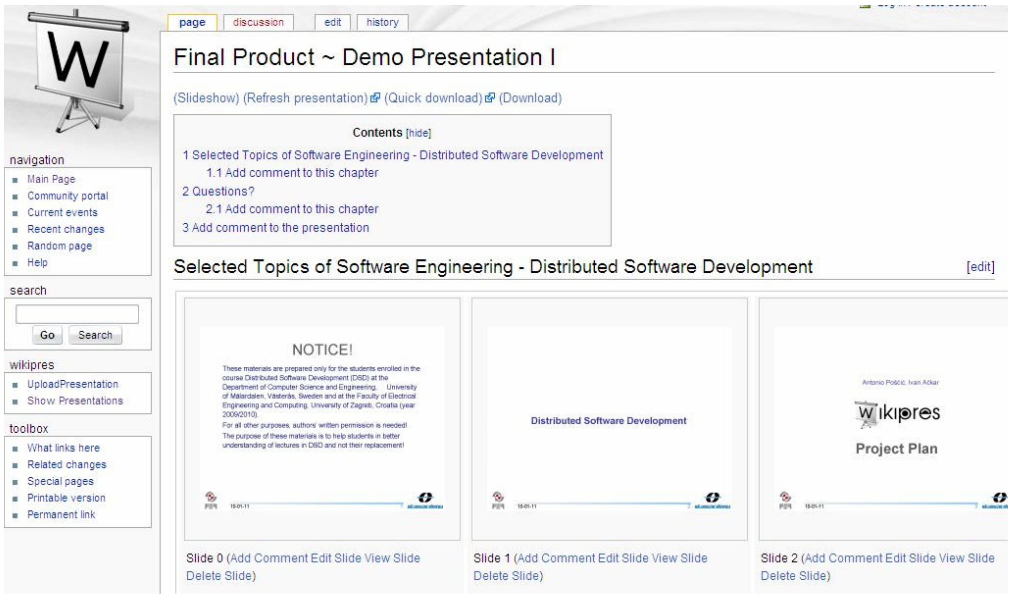
The most typical use case for such collaborative work is described as follows: The teaching staff publishes the course presentation slides, through the system which supports collaborative activities. The users (staff and students) view the presentations in different forms, as a slideshow, a table of slides, or slide by slide. Upon finding the part which is unclear, contains mistakes, or can be improved, the users can either comment or directly edit the content (if permitted by the staff). In case of commenting, other users see the comments together with the additional content. In case of editing, the content is automatically refreshed. The teaching staff can additionally export the presentations and view them offline, including all the comments and changes made by students.

The prototype implementation of such application is described in the following section.

B. *WikiPres application*

In order to enable teachers and students to create, comment and edit lecture slides, the *WikiPres* application was implemented. The characteristics of *Wiki* in general, such as collaboration possibilities, quick content editing, focus on content instead on design, and revision history, provide a good basis for our use case. The application is designed as a *MediaWiki* extension to build upon this most popular *Wiki*-based system. Some of the *WikiPres* features are:

- presentation importing in *ODP* or *PPT* format,
- detection of presentation chapters based on master slides and slide layouts: useful in situations when presentations have a large number of slides,
- categorization of presentations and per-category permission granting,
- commenting the whole presentation, presentation chapters and individual slides,
- editing the slides in *WikiMedia* markup,
- presentation and individual slide overview (shown in Figure 1),
- presentation exporting in either *ODP*, *PPT*, or *PDF*, with an option to also export the comments for the slides, chapters, and presentation,
- slideshow: a feature that enables a slideshow view of the presentation (adaptive to the screen resolution).


page discussion edit history

Final Product ~ Demo Presentation I

(Slideshow) (Refresh presentation) (Quick download) (Download)

Contents [hide]

1 Selected Topics of Software Engineering - Distributed Software Development
  1.1 Add comment to this chapter
2 Questions?
  2.1 Add comment to this chapter
3 Add comment to the presentation

Selected Topics of Software Engineering - Distributed Software Development [edit]

NOTICE!

These materials are prepared only for the students enrolled in the course Distributed Software Development (DSD) at the Department of Computer Science and Engineering, University of Mälardalen, Västerås, Sweden and at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia (year 2009/2010).

For all other purposes, authors' written permission is needed!

The purpose of these materials is to help students in better understanding of lectures in DSD and not their replacement!

Slide 0 (Add Comment Edit Slide View Slide Delete Slide)

Distributed Software Development

Slide 1 (Add Comment Edit Slide View Slide Delete Slide)

Antonio Pobčić, Ivan Ačkar

Wikipres

Project Plan

Slide 2 (Add Comment Edit Slide View Slide Delete Slide)

Figure 1. Presentation overview in WikiPres

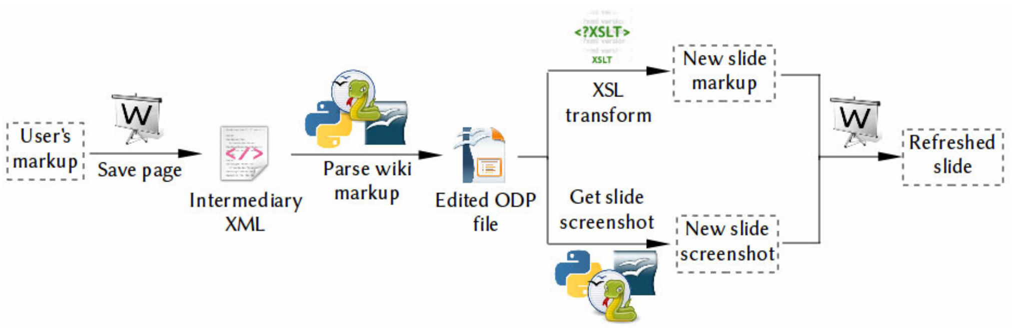


Figure 2. WikiPres data flow

After uploading the *ODP* or *PPT* presentation, the presentation is divided into chapters based on algorithms which analyze use of master slides and slide layouts. Figure 2 shows the process of converting content from slides to *WikiMarkup* and back. The presentation content is converted into an intermediary *XML*, and converted to *WikiMarkup* with the help of *XSL* transformations and *PyUNO* bridge for *OpenOffice.org*, written in Python [3]. The slides become the regular *Wiki pages*, with the additional pages for the presentation overview and chapters. The users can browse the presentation using thumbnail images, generated using *ImageMagick*. If the slide is edited, its content is compared to the previous version to preserve as much slide formatting as possible for future exporting. The new slide screenshot is generated, so the users can see the changes immediately. In addition, the *XML* content of the slide in *ODP* format is generated upon editing, so that the export process is quick even for large presentations.

IV. SMART ASSIGNMENTS VERIFICATION

Assignments in the *Open computing* course are designed to illustrate concepts introduced in the lectures. Assignments cover a wide range of technologies to illustrate the openness of protocols and technologies underlying the modern web. When all assignments are completed, they come together to form a small dynamic web site. In general we can divide the assignments in two main groups – illustrating client- and server-side technologies (during the first two course cycles). Final assignments focus on integration of information sources and enhancing the experience for fictitious users of the web site.

Introductory assignments require students to create a few skeleton web pages (client-side) which are then augmented with dynamic web functionalities towards the end of the course. Server-side oriented assignments focus on integration of various technologies: *PHP* and *Java Servlets* for backed processing and *XML*-related technologies to transfer and query data. Throughout the course, lectures are used to introduce students to new concepts, briefly illustrating related technologies, while assignments require students to put these technologies to use. The lack of experience with newly introduced technologies and detailed configuration, required for some assignments, have been the most common source of problems for students.

To help students with their assignments *ORVViS*, an assignment verification system linked with *Moodle* LMS, was developed. After students submit their assignments to LMS, *ORVViS* verifies the submissions and reports errors to students, to help them fix the errors before the assignment is due. The introduction of *ORVViS* greatly reduced a number of common problems student had with assignments, such as invalid *HTML*, *CSS* or servlet configuration files, therefore reducing the number of assignment-related questions in forums, often posed in odd hours of the night. *ORVViS* has been tailored to provide separate validation for each file type the submission requires. The

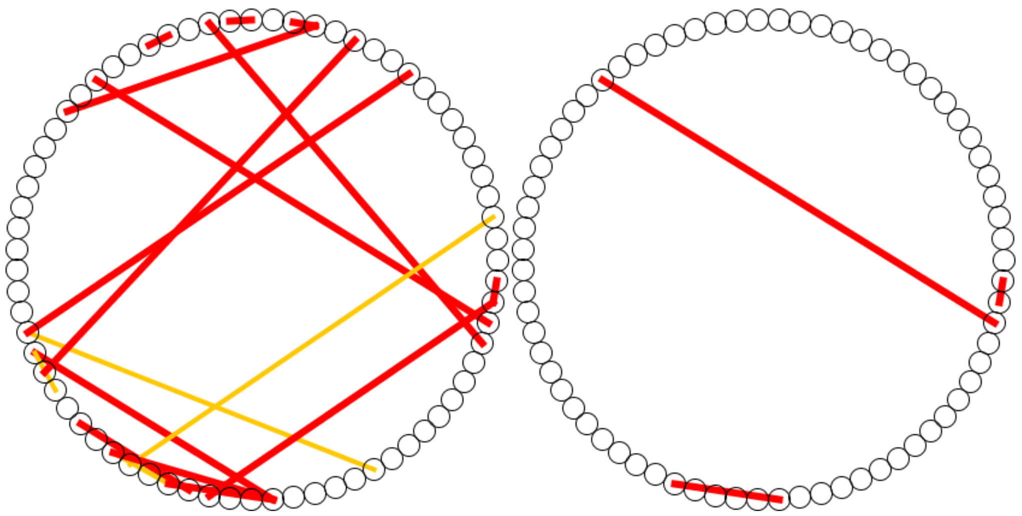


Figure 3. Detected submission similarity with threshold levels of 20% (left) and 80% (right).

reports students receive contain aggregated validation reports obtained from common validation tools (*HTML* and *CSS* validators, *Java* compiler and such). The description for each assignment also lists useful validation tools that students can use themselves. However, before *ORVViS* was introduced, most students would first post questions on the course's forums and then try validating their files, or would even submit invalid files to LMS.

In addition, *ORVViS* prepares submissions for plagiarism detection tools [5]. Since the introduction of *ORVViS*, we have observed a gradual decline of (detected) assignment plagiarism, as *ORVViS* helped students to resolve most common problems and successfully complete assignments with less effort.

A. Assignment verification

A typical use case for our students follows. Student should first read the assignment description and related documents, and use provided code snippets or examples as a head start for her own assignment solution. After she had created a solution that meets all of the requirements, the student will submit it to LMS. In a short while, she will have received an e-mail message containing submission validation results from *ORVViS*, and correct reported errors. After several iterations to remove reported errors, the submission can be marked as final in the LMS. The student will then present her solution to teachers on designated laboratory time and answer questions related to the assignment.

Automated assignment verification tool *ORVViS* consists of two main units: verification core and *Moodle* LMS integration modules. In order to seamlessly introduce ORVViS to students, verification was integrated with LMS using *Moodle* APIs to fetch student submissions and create an *ORVViS* administration block plugin for LMS pages. The verification part of *ORVViS* is configured to run verification tasks on specific assignments from LMS. Each task lists verification plugins that will be run for each file type submitted. Tasks are run periodically, fetching newly submitted assignments and reporting to students.

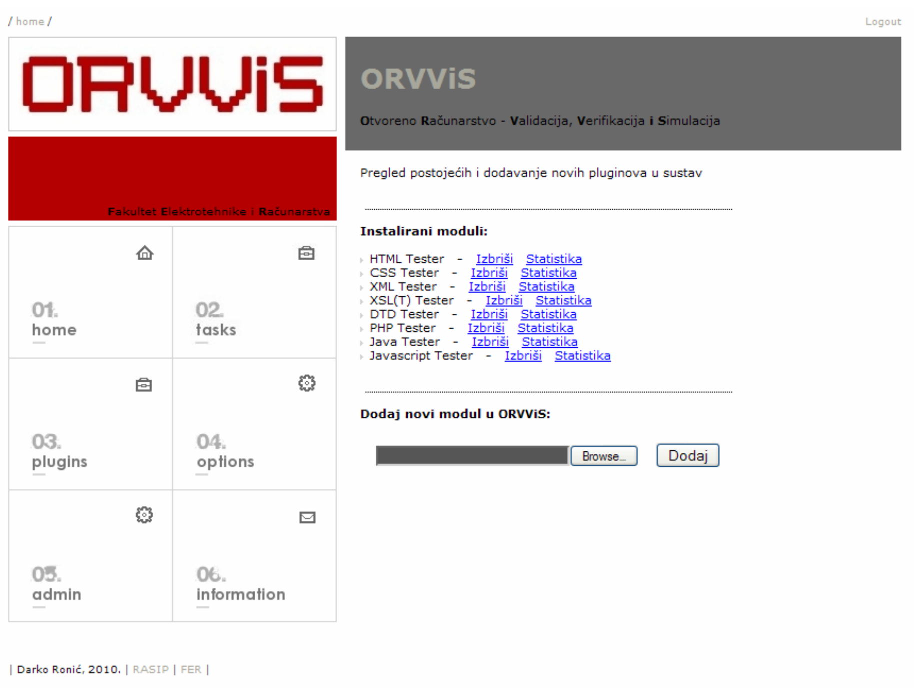
When the assignment deadline is reached, a cumulative report is created for the course administrators. These reports contain bulk overview statistics for all submissions (number of submitted assignments, number of successfully validated assignments and such). If configured, administrative reports can also contain reports of plagiarism detection tools (Sherlock plagiarism detection based on digital signatures is used [4]).

However convenient, fully automated plagiarism detection is not appropriate for all assignments as large chunks of source code are meant to be reused by students in some assignments. *ORVViS* prepares submission files for supervised analysis. Files from student submissions are renamed and organized by their type (i.e. *CSS* files are separated from HTML files) to allow detailed analysis using other tools such as (identically named) *Sherlock* from the *BOSS* package [5]. Setting an adequate threshold for detection requires manual review of detected submission pairs to detect false positives occurring at lower threshold levels, as seen on Figure 3.

B. ORVViS application

The two parts of the *ORVViS* application are developed to act as background processes administered with a simple administrative interface (illustrated on Figure 4). *ORVViS* periodically polls *Moodle* LMS for new submissions using *Moodle* API, downloads the archives with student submissions and verifies the contained files using configured validation plugins. Plugins for validation have been developed using freely available validators for *HTML*, *CSS*, *XML*, *XSL(T)*, *DTD*, *PHP*, *Java* and *JavaScript* files, such as *Tidy* (*HTML*), *DOM* (*XML*, *XSL*), *JavaScript Lint* (*JavaScript*) and *PMD* (*Java*). *ORVViS* administrator configures associates plugins with specific files in student submissions using filename patterns.

After validation of student submissions is complete, each student receives a detailed report listing validation status for each file and warning and error messages reported by the corresponding validator. Some validators have been set to be more sensitive and report more detailed warnings than a typical compiler would. This has proven effective in cases where students used newer compiler and runtime versions (e.g. of *Java*) than those available on our servers, as additional warnings would give students a pointer where to start looking for problems.



ORVViS

Otvoreno Računarstvo - Validacija, Verifikacija i Simulacija

Pregled postojećih i dodavanje novih pluginova u sustav

Instalirani moduli:

- HTML Tester - Izbriši Statistika
- CSS Tester - Izbriši Statistika
- XML Tester - Izbriši Statistika
- XSL(T) Tester - Izbriši Statistika
- DTD Tester - Izbriši Statistika
- PHP Tester - Izbriši Statistika
- Java Tester - Izbriši Statistika
- Javascript Tester - Izbriši Statistika

Dodaj novi modul u ORVViS:

| Darko Ronić, 2010. | RASIP | FER |

Figure 4. Task administration in ORVViS

V. LESSONS LEARNED

This section outlines some of the non-technical issues observed while testing the prototypes of *WikiPres* and *ORVViS* applications.

The main issue related to collaborative content editing is the students' motivation. Although initial surveys carried out among students of various program years showed the need and the will for such way of learning [6], in reality it is still hard to motivate them for such a collaboration. This year, the teaching staff did not insist on using *WikiPres*, as it was still a prototype; we rather wanted to find out would the students use it without external encouragement and strictly defined rewards, in terms of course points or similar. Several students tested it, but due to different obligations on this and other courses, for them it was more important to participate in obligatory activities which were clearly awarded by course points. Another possible reason was that the students are generally not used to give and defend their opinions in an open setting where the teaching staff is also present. As an example, students are not used to publicly point the errors of the official lecture notes as a part of the course activities. This open expression of opinion should be additionally encouraged.

The teaching staff should be prepared to listen to the students' comments and work closely with them in improving the current content without prejudice. This can sometimes require the staff to change the approach to education, with a goal of changing the teaching perspective from transmissive to developmental [7]. In the future years, we plan to encourage students by taking these activities into account when grading.

The students' awareness of the proper *Wiki* work ethics and their responsibilities when collaborating on editing the course content is an additional issue. As we strive to be open, monitoring (and sanctioning) the students should be on a minimal level. In the extreme cases in which damage to the content has been made, the system provides a way to revert to the old versions of presentations.

ORVViS verification focuses mainly on syntactic validation of submitted files and plagiarism detection. Assignment requirements are specified loosely with the intent of allowing a large degree of freedom for students to play with. When students present their solutions, it is fairly easy for teachers to inspect its correctness with respect to these requirements. After *ORVViS* was introduced to the course, we have seen an increase in the ratio of valid submissions from 61% to 89% for two slightly modified introductory assignments. During assignment presentations, we have observed that additional time students gained by not having to trace common errors gave students an opportunity to better understand which parts of the assignments are dictated by standards and which parts they need to produce. Automated verification has a downside as students tend to overly depend on *ORVViS* to report on all but the most serious errors. After *ORVViS* has been unavailable for a few days due to server failure, the ratio of valid submissions dropped to 43% with many minor errors and warnings still existing in solutions during the system outage. Part of the students didn't even try to use independent validation tools instead of *ORVViS*.

Not surprising, unlike *WikiPres*, we had to invest very little to motivate our students to use *ORVViS* and follow suggestions from its reports, as they could clearly see the benefits of such a system. Initial *ORVViS* versions tested by students were not completely integrated with the Faculty's LMS as this was a risk to the production system. For testing, students were asked to upload their submissions to both the "*official*" and "*ORVViS*" LMS installations, which they did enthusiastically – for as long as they considered validation reports to be useful, but stopped to do so as soon as we experienced problems with validation plugins. This experience strengthened our belief that the verification system needs to be integrated with the official LMS and invisible to the students.

VI. CONCLUSION AND FUTURE WORK

With regard to technical issues in *WikiPres*, users proposed easier slide comments handling, and navigation more appropriate to the presentation format. While addressing these, one should be careful about integrating the different paradigms of content formats: presentations on one side, and the *Wiki* way of working with content on the other. For future releases, implementation of plugins for the most common office suite presentation software (*OpenOffice.org Impress*, *Microsoft PowerPoint*) is planned, to aid in uploading and conversion of presentations conversions. Integration of *WikiPres* with LMS would be useful, as the students would have a common place to access both the versions of the content: the initial one and the one improved as the course advances. Researching of the ways which would allow recommending of learning resources to be included in the content is already being carried out. There are other interesting topics, pointing in the direction of increased students' activity as a combination of online and face to face teaching. For instance, asking questions or providing related content during the lectures through the *Twitter* service, as the majority of students follows the lectures while using their laptops, etc.

Most of the ideas for *ORVViS* improvements came from the teaching staff, as students have very little contact with the system itself and were generally satisfied with verification reports received. One of suggested improvements was to fully integrate *ORVViS* administration in *Moodle* LMS, to avoid the need to administer verification tasks and plugins in a separate interface. We also plan to implement additional plugins to verify semantic properties of student submission, instead of concentrating mostly on syntax. *Open computing* course focuses on the benefits of clearly defined interfaces on various levels of the application to achieve openness. Test cases checking various properties of student solutions can be used for automated testing, and distributed to students to give them the opportunity to see the benefits of the approach of *reflection in action* over the more common *trial and error* [8]. The heterogeneity of technologies used in our assignments makes it difficult to further integrate *ORVViS* into development tools that students use (as demonstrated by [9]), but it is a path we intend to investigate.

ACKNOWLEDGMENTS

We would like to thank our students Ivan Aćkar, Antonio Pošćić, Zvonko Žibrat and other students in the *WikiPres* project group on Distributed Software Development course held jointly with Mälardalen University. We would also like to thank our student Darko Ronić and other students under his supervision for their work on *ORVViS* application.

REFERENCES

[1] S. Tomić,.; K. Zimmer, M. Žagar, V. Paunović, I. Voras, "Living The E-Campus Dream", *Proceedings of the EDEN Conference, A. Szucs and I. Bo, Vienna, Austria: European Distance and E-Learning Network*, pp. 644-650, 2006

[2] S. d'Antoni,. C. Savage: Open Educational Resources: Conversations in Cyberspace, EU Report, 2009.

[3] OpenOffice.org Wiki.: PyUNO bridge. http://wiki.services.openoffice.org/wiki/PyUNO_bridge

[4] Sherlock: Plagiarism Detector. http://www.cs.su.oz.au/~scilect/sherlock/

[5] University of Warwick.: BOSS Online Submission System. http://www.dcs.warwick.ac.uk/boss/

[6] I. Bosnić, A. Pošćić, I. Aćkar, Z. Žibrat, M. Žagar, "Online Collaborative Presentations", *Proceedings of the 32nd International Conference on Information Technology Interfaces - ITI 2010*, Cavtat/Dubrovnik, Croatia: pp. 1-6,. 2010,

[7] D.D. Pratt, "Five Perspectives on Teaching in Adult and Higher Education", Krieger Publishing Company, 1998.

[8] S.H. Edwards: "Using software testing to move students from trial-and-error to reflection-in-action", ACM SIGCSE Bulletin, vol. 36, 2004, p. 26. doi:10.1145/1028174.971312

[9] A. Allowatt, S. Edwards, "IDE Support for test-driven development and automated grading in both Java and C++", *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange - eclipse '05*, 2005, pp. 100-104.

AUTHORS

Ivana Bosnić is a PhD student and research/teaching assistant at the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering, Unska 3, 10000 Zagreb, Croatia (e-mail: ivana.bosnic@fer.hr).

Marin Orlić is a PhD student and research/teaching assistant at the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering, Unska 3, 10000 Zagreb, Croatia (e-mail: marin.orlic@fer.hr).

Mario Žagar is a tenure professor of computing at the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering, Unska 3, 10000 Zagreb, Croatia (e-mail: mario.zagar@fer.hr).

This work is supported in part by the Croatian Ministry of Science, Education and Sports, under the IT project *e-learning Now*, and research project *Software Engineering in Ubiquitous Computing*.

Submitted, October, 22, 2010. Published as resubmitted by the authors on November 22nd, 2010.