Incremental Development of E-Learning Systems for Mobile Platforms

Milena Frtunić[1], Leonid Stoimenov[2] and Dejan Rančić[3]

Abstract – **In this paper incremental development of e-learning system for mobile devices is presented. The benefits of incremental model are discussed and reasons for choosing this model for developing e-learning and m-learning software is given. Further, this approach is explained on the MoodleQuiz application, developed for taking the Moodle quizzes on Android mobile devices.**

Keywords – **E-learning software system, M-learning, Moodle quiz, Incremental development, Android development.**

I. INTRODUCTION

Advancement of technology in the past decade has resulted in a great progress in mobile technology. Mobile hardware and the networks that support them became more powerful, more dynamic and more affordable. Because of that, number of their users is growing rapidly, and it is expected that in the next few years most of the users will be accessing the Internet by using their mobile devices and smartphones. Due to this, the need for all information on the Internet to become available on the mobile devices has emerged.

Since, most of the learners today are likely to have mobile devices to access the Internet with them all the time (at home, at school, on public transportation, at work etc.), it is reasonable to expect that they would like to use those devices for every need they have on the Internet [1]. This means, that it is critical to shift to the mobile devices, not only regular information, but also e-learning systems. In the past few years, many countries came to the similar conclusion and made steps in developing software that will support informal learning and run on the mobile devices such as tablets and smartphones. Due to that, m-learning is gaining more popularity every day.

Education sector usually requires software delivery in minimum possible time [2]. Delays in delivery of software might not be acceptable to most education organizations. Moreover, misinterpreted, missing or incomplete requirements might result to errors in final product, which can lead to a huge problem for education institutes. In order to provide efficiency of development and reducing possibility of errors, incremental development of the software is recommended.

In the next part of this paper, the choice of incremental development will be further discussed. After that we will show incremental model on example of MoodleQuiz [3] application, developed for solving Moodle Quiz on Android mobile devices.

[1]Milena Frtunić is with the Faculty of Electronic Engineering, University of Nis, A. Medvedeva 14, Nis 18000, Serbia, e-mail: milena.frtunic@elfak.ni.ac.rs.

[2]Leonid Stoimenov is with the Faculty of Electronic Engineering, University of Nis, A. Medvedeva 14, Nis 18000, Serbia, e-mail: leonid.stoimenov@elfak.ni.ac.rs.

[3]Dejan Rančić is with the Faculty of Electronic Engineering, University of Nis, A. Medvedeva 14, Nis 18000, Serbia, e-mail: dejan.rancic@elfak.ni.ac.rs

II. INCREMENTAL DEVELOPMENT

Developing e-learning software can be very challenging. The resulting product has to be system that will be well received not only by students, but also by teachers. This can be difficult because not everyone is used to informal learning methods and using e-learning systems for process of teaching and testing students. For that reason, it is very important to develop system that teachers will understand and find useful in the teaching process.

Situation can be even more difficult for developing these systems for mobile platforms. Main reason for that is specificity of these devices. Application developed for mobile devices and smartphones have to be more personal, more interactive and fast, so that users find it interesting and useable. Since mobile devices and smartphones have small screen sizes this can be challenging [4]. The best way for developing those applications is incremental development model. Main reason for this choice is that after every increment, the developers can have a feedback from the customers and make changes in the product by their comments.

The main characteristic of incremental model is that the whole system can be developed in increments, starting with the most important requirements at the beginning. When developing m-learning systems, that can be very useful. Reason for this is that acceptability among the teachers and students can be tested after every increment. Moreover, active user involvement throughout the product's development and a very cooperative and collaborative approach can help in ensuring that expectations are effectively managed. Also, that can help in identifying any issues in early stages and make it easier to respond to change.

On the example of m-learning software development, this means that educators can get the feeling how the system is going to work, monitor the progress and alert if the system is not working as it is supposed to. Furthermore, after every increment educator can test the complete requirement and get the reactions from student and see how they feel about using the application on regular bases in their studies. Additionally, developers can get the feedback about how usable the application interface is, in the early stages of application development. The reaction received from the students is

extremely valuable, because the project will not be successful if the students don't receive the application well and if they find it difficult to use.

III. MOODLE QUIZ

MoodleQuiz, application developed for solving Moodle quizzes [5] on Android mobile devices, was developed using incremental model because of the previously explained reasons. MoodleQuiz is part of the "Infrastructure for electronically supported learning in Serbia" project and it should provide possibility for participants to solve quizzes on portal that will be developed within this project. In development process, testing of the application will be done on Moodle portal that is used on Faculty of Electronic Engineering at University of Nis.

Since this system is part of the much bigger project, this paper covers only first stage of application development that was done in two increments. After every increment unit testing was done.

A. *First increment*

First increment included development of the Moodle plugin – Moodle mobile quiz plugin - that enables third party application to access the quiz information and provides support for solving Moodle quizzes on third party application.

Also, in this increment the foundation of the application was set and communication with the Moodle server was established. The deployment diagram of the application is given in the Fig. 1.

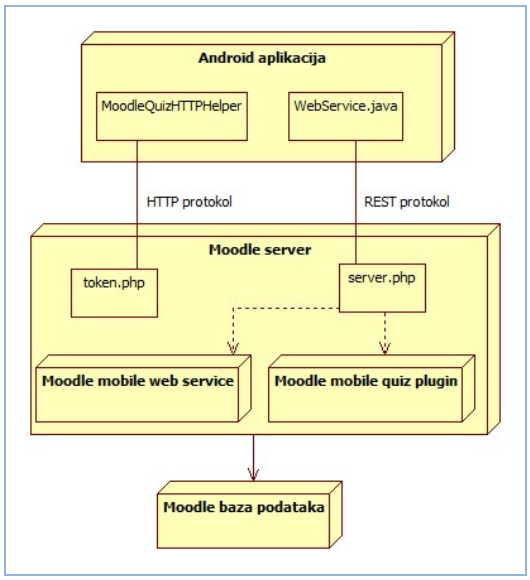

Fig. 1. Deployment diagram

As it is shown on the deployment diagram, application communicates with the Moodle server using HTTP and REST protocol. HTTP protocol is used for logging user on the Moodle system and providing tokens for using Moodle mobile web service and Moodle mobile quiz plugin. The example of the code that is used for providing tokens for the user is given in the Fig. 2. Function *String getToken(url)* belongs to *MoodleQuizHTTPHelper* and is used for providing tokens. If the output from this function is empty token, the user can not continue forward with using the application and logging is not successful.

```
String url = server_url+"/login/token.php?username="
            + usrUri + "&password="
            + pwdUri + "&service=moodle_mobile_app";
MoodleQuizHTTPHelper tokenRequest = new MoodleQuizHTTPHelper();
token = tokenRequest.getToken(url);

url = server_url+ "/login/token.php?username=" + usrUri
            + "&password=" + pwdUri
            + "&service=mobile_quiz_plugin";
MoodleQuizHTTPHelper tokenRequest2 = new MoodleQuizHTTPHelper();
tokenQuiz = tokenRequest2.getToken(url);
```

Fig. 2. Example of providing tokens for user

Furthermore, in this increment basic functions for choosing course and quiz that he wants to take in the chosen course were created. In the implementation, this part was done by using methods from *WebService* class, created within this application. This class contains all methods for establishing communication with the Moodle server. In the Fig. 3 the example of code from this class for getting all courses in which the user is enrolled is given.

```
String serverurl = MainActivity.getServerUrl()
    +"/webservice/rest/server.php" + "?wstoken="
    + user.getToken() + "&wsfunction=";
WebService webService = new WebService(CourseActivity.this);
webService.getSiteInfo(serverurl, user);
```

Fig. 3. Example of getting courses

In the example in Fig. 3, *WebService* method *getSiteInfo(String serverurl, User user)* is called. This method, as all others, calls Moodle web service and appropriate function. The communication with Moodle system is done by using REST protocol and the response is received in JSON format. The example of such communication is given in Fig. 4.

```
url = serverurl + "moodle_webservice_get_siteinfo"
            + "&moodlewsrestformat=json";
String urlParameters = "serviceshortnames[0]=moodle_mobile_app";
odgovor = getWebServiceResponse(url, urlParameters);

JSONObject jinfo = new JSONObject (odgovor);
String firstname = jinfo.getString("firstname");
String lastname = jinfo.getString("lastname");
String userid = jinfo.getString("userid");

user.setFirstname(firstname);
user.setLastname(lastname);
user.setId(Integer.parseInt(userid));
```

Fig. 4. Example of communication with Moodle server

This increment covered implementation of the part that provides basic information about the quiz that user wants to

solve. Also, it provides registration to the quiz and creation of the attempt. Creation of the attempt is in fact creation of the quiz attempt on the Moodle server as well as pulling all questions for the attempt along with the answers for those questions included. Additionally, in this process quiz setting are received from server. This was done in a similar way as explained in Fig. 4, only difference being that for this communication, new developed Moodle plugin is used.

Despite the requirement to enable four types of questions to work in the application: true/false questions, questions with short answers, multiple-choice questions and matching questions. First increment covered only two types: true/false and short answer questions. This was done in order to get the feedback from teachers and students about how they like the application, before implementing all types of questions. Moreover, in this increment the design for application was done, too. Reason for this decision is that application design is very important when creating application for mobile devices, mainly because of the small sizes of displays.

After the first increment was published, few teachers and students tested the application and gave their feedback on their opinions and impressions. Since the comments were positive and users were satisfied with the progress of the application development, the development of the second increment began.

B. Second increment

In second increment focus was on developing other questions that were required in the specification for the application. That meant implementing multiple-choice questions and matching questions. Since every question has different way of presentation and evaluation, every question is implemented as a class that extends abstract class *Question* with methods common for all types. In Fig. 5 this organization is shown in class diagram.

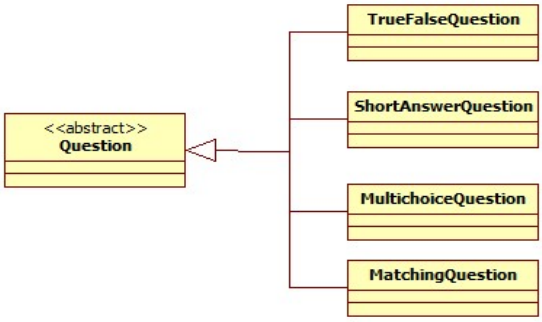

Fig. 5. Question class diagram

Class *Question* beside functions common for all types of questions, contains abstract methods:

- *void prikaziPitanje(LinearLayout ll, LinearLayout llt)* – is used for displaying question during the quiz attempt;
- *void zapamtiOdgovor(LinearLayout ll)* – saves question answer;
- *void oceniPitanje()* – counts points that student won on a question;
- *String odgovorZaServer()* – preparing an appropriate information for the Moodle server, i.e. data that will be entered into a database to make it consistent again after the end of the quiz attempt;
- *String getChoice()* – returns data that will be entered into the table *question_attempt_steps_data*;
- *void prikaziRezultat(LinearLayout ll, Context c)* – is used for displaying question result after the attempt was finished, in the mode where student is reviewing his results accomplished in the attempt.

Beside implementing required types of questions, in this increment additional components were developed, such as parts for evaluating quiz attempt, displaying quiz results to the user and updating Moodle database so that course administrator can review the quiz results on Moodle system.

Submitting the quiz attempt is done immediately after the quiz is finished. All questions and answers the student gave, together with points won on each question are sent to the Moodle server by using *WebService* class method *submitQuiz*. This method calls *update_finished_attempt* function of Moodle mobile quiz plugin. Example of the code that sends the results to the server is given in Fig. 6.

```
for(int i=0;i<questions.size();i++){
    q = q + "&quid["+i+"]=" + questions.get(i).getId();
    a = a + "&answers["+i+"]=" + Uri.encode(questions.get(i).odgovorZaServer());
    p = p + "&qupoints["+i+"]=" + questions.get(i).getOcena();
    qutype = qutype + "&qutype["+i+"]=" + questions.get(i).getType();
    studchoice = studchoice + "&studchoice["+i+"]="
            + questions.get(i).getChoice();
}

String url = serverurl + "local_wstemplate_update_finished_attempt"
        + "&moodlewsrestformat=json";
String urlParameters = "attemptid="+quiz.getUniqueid()
        +"&userid="+user.getId() +"&timefinished="+preostaloVreme;
urlParameters += "&points="+quiz.getMark() + q + p + a + qutype + studchoice;
String odgovor = getWebServiceResponse(url, urlParameters);
```

Fig. 6. Code for updating results on Moodle server

The final results of application achieved after two increments are shown in Figs. 7 and 8. Fig. 7 presents an example of matching question during the quiz mode. In the Fig. 8 is presented a review of the results after the quiz is finished. In this example, results of two multiple-choice questions where student gave wrong answer on both questions are shown.

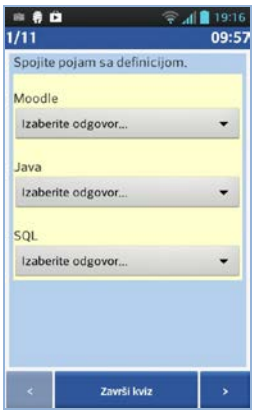

Fig. 7. Example of matching question

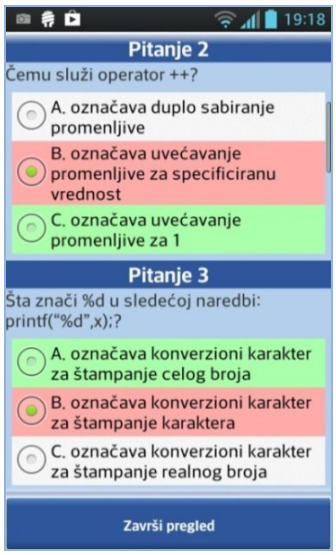

Fig. 8. Example of quiz results

After second increment was published, the result was MoodleQuiz application that supports four most commonly used types of questions that may occur in the quiz and reviewing quiz results. The application is fully functional and works on all Android Mobile devices that support 2.2 or higher version of Android operation system. It has a simple design and application layout is optimized for use on small sized screens.

IV. Conclusion

After second increment was finished, MoodleQuiz application was presented to teachers and professors. Their reaction to the application was positive. They found the MoodleQuiz usable and interesting for using not only for testing students' knowledge on the exams, but also for getting feedback from students in the lectures on how much the students understood the lecture. They feel that student will gladly use this application during the preparation of the exam for verifying their knowledge.

Due to the positive reactions on the application, preparation for MoodleQuiz testing began. The plan for testing to be done with students attending second year at Faculty of Electronic Engineering at University of Nis and are enrolled on course Database systems. Testing will be done on more than 120 students that are taking this course in spring semester 2014. The goal of this testing is to see:

- how many students will download the application;
- how many of them would use it for testing their knowledge while preparing exam;
- how many students would prefer doing the quiz on mobile device rather than on a computer in the classroom;
- what is their overall impression of the application;
- how many of all tested students would like to continue using the application on regular bases;
- how do they like usability and user interface.

The plan is for students to test the application at the end of the semester. Depending on the results at the end of the application testing, plan is to continue application development. It will be done in few increments. Each increment will include development of one question type that Moodle system supports and that was not covered in first two increments. At the end of all increments all types of questions will be covered, not only the most used ones.

Further, these increments should cover upgrade of *Moodle mobile quiz* plugin. The upgrade should enable better tracking of students' actions during the quiz. Moreover, third increment should enable automatic generation of tokens for *Moodle mobile quiz* plugin, so that course administrator won't have to generate separate token for each student that wants to use the application.

If the application comes to life, plan is to expand the application with other modules that Moodle system offer, so that students will be able to view other course details, and not only quiz information.

Acknowledgement

The research presented in this paper was funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia as part of the project "Infrastructure for electronically supported learning in Serbia" number III47003.

References

[1] A. Tsinakos, *State of Mobile Learning Around the World*, Global Mobile Learning Implementations and Trends, 2013.

[2] V. Gupta, D. S. Chauhan, K. Dutta, *Incremental development & revolutions of E-learning software systems in education sector: a case study approach*, Human-centric Computing and Information Sciences, 2013.

[3] M. Frtunić, M. Bogdanović, L. Stoimenov, "Moodle Quiz on Android Mobile devices" YU INFO 2014, pp 7-12, Kopaonik, Republic of Serbia, 2014.

[4] P. Pocatilu, M. Doinea, C. Ciurea, "Development of Distributed Mobile Learning Systems", Recent Researches in Circuits, Systems, Electronics, Control & Signal Processing, pp 196-201, Athens, Greece, 2010.

[5] Moodle quiz, http://docs.moodle.org/25/en/Quiz_module.