Design and Development of a Mobile Classroom Response System for Interactive Problem Solving*

M. Muztaba Fuad
Department of Computer Science
Winston-Salem State University
Winston-Salem, NC 27110, USA
fuadmo@wssu.edu

Debzani Deb
Department of Computer Science
Winston-Salem State University
Winston-Salem, NC 27110, USA
debd@wssu.edu

Abstract— **It is common for students to multi-task and use their mobile devices while in class for studying or other activities. This research aims to leverage this situation by developing a mobile classroom learning software to help students solve interactive problems in their mobile devices in order to improve their class engagement and problem solving skills. This paper presents the design and development of the mobile classroom response software to communicate, collaborate and evaluate in-class interactive problem solving activities using mobile devices. The software facilitates various pedagogical approaches that reported to enhance student learning. The software is designed to be easy-to-use and maintainable. MRS is extensible and can render interactive problems developed by third party developer. Software quality matrices for the developed code and the user interface are presented to justify above stated objectives.**

Keywords-Mobile Learning Software; Software Architecture, Client-server System, Extensibility.

I. INTRODUCTION

Mobile technology has brought tremendous potential and opportunities for educators to enable and deliver learning in ways that could not have been accomplished before. There have been an increasing number of studies related to the research and development of learning software intended for mobile computing devices. Some of these studies focus on supporting several interesting pedagogical approaches while utilizing mobile learning software such as slide annotation, collaborative note taking, student submissions, grading, polling etc. This research envisions utilizing mobile learning environment to help students solve interactive problems in order to improve their class engagement and problem solving skills. Toward this goal, this paper present the design, development and planned incorporation of the mobile classroom response software (MRS) designed to communicate, collaborate and evaluate in-class interactive problem solving activities using mobile devices.

MRS is a client-server software that allows the faculty to dynamically prompt the students with interactive problems synchronized with the lecture material in their mobile computing devices (Clients). Students are able to actively interact with the problem and send their answer back to the faculty computer (Server). MRS then performs grading of the exercises automatically, by comparing the student made sequence of steps with the correct sequence of steps. The other important features supported by MRS are immediate and context-sensitive feedback, anonymous question, polling, summarized grading etc. Currently MRS software and associated problem solving activities are being deployed in CS and IT courses at Winston-Salem state University (WSSU). By adopting MRS in the classroom, this research expects to increase student engagement and improve their problem solving capabilities and therefore prepare them better to enter the computing workforce. MRS is designed to be user friendly and extensible so that faculty and students can use the system with ease and can extend it to any domain.

II. BACKGROUND AND RELATED RESEARCH

To improve student learning in the STEM disciplines, traditional pedagogical approaches are not enough to transfer critical knowledge to students. A feedback driven evidence based teaching and learning technique has to be devised and implemented to improve student retention rates in the STEM discipline. In that regard, evidence-based instructional practices were incorporated in a sophomore-level CS course for the last few offerings, where at the end of a lecture, students were immediately prompted with in-class problem solving activities. Intervened student's performance data and their response to this pedagogy reveal the potential of this approach in order to enhance student learning. This finding encourages us to extend this model by scattering the questions/problems throughout the class period by synchronizing with the content covered. Observation about student's frequent use of mobile devices during class further inspires us to extend the above idea of asynchronous problem solving to mobile devices to engage students more to class activities.

Different studies [1]-[2] have found the benefits of using mobile devices in classrooms and in recent years, there has been a plethora of work [3]-[4] etc. performed to incorporate mobile devices in classrooms. There are also several commercial products [5]-[6] etc. for different mobile platforms that provide similar functionalities. Classroom Presenter [7], Ubiquitous Presenter [8] and DyKnow [9] are notable research initiatives that utilize tablets to create more active, student-centered lecture environment while supporting various effective pedagogies. However, there are distinct differences

* Supported by NSF fund # 1332531

between the MRS and similar systems. Most importantly, MRS facilitates interactive problem solving. In STEM courses, students need to actively solve problems by interacting with the problem in a hands-on approach. Students cannot develop skills such as synthesizing a problem and critical thinking only by using multiple-choice or true-false questions. We argue that by presenting the problems as interactive entities, where students can actively play with the problem; student's critical thinking and problem solving skills can be improved.

A. Interactive Problem

An interactive problem is one, where students have to devise the answer following a set of steps and by following a particular algorithm/process. In each step, students have to make key choices that will have impact on the next step of the interaction. During these interaction steps, students can go back and forth and change their answer. This will allow them to see what is the affect of different selection on the result and how every piece fits together. Problems can be started bottom up or at the middle to give students different perspective on the problem and assess their problem solving skills. Only after the student traverse each of the steps or the allotted time to answer a problem runs out, the results of their interactions performed at each step are then sent back to the server as the answer. Each problem has a rubric that not only grade final answer but also partial answers to gauge student's problem solving skills and thinking models.

III. SYSTEM DESIGN

The MRS software is designed as a client-server application. Client device needs the corresponding client app installed in them to interact during the class. The server (or the faculty computer) hosts the questions, manage users and process the results for display and for grade calculation purposes. It also has the required data analysis component to tabulate user responses and produce easy to interpret reports.

A. Server

Figure 1 shows the lifecycle of the server. The server is designed as a multi-process, multi-threaded entity to satisfy simultaneous invocation from users and to provide real time response to in-class activities. Usually when a user initiates a check-in to the system, the server validates the identity of the user and set a role for that specific user. Roles are basically privilege levels that allow a user to interact with the system with certain accessibility. The client app also allow students to send anonymous feedback/questions to faculty and vote on questions that faculty will choose to review at the end of the class. Separate thread of the server constantly monitors whether students wants to initiate a feedback/question session. In that case, it sends the current pool of feedback/ questions to the corresponding client. Once a response from the client is received, it is matched with the current pool and if a match is found, the priority of that entry is increased. Otherwise a new entry is created with the newly initiated feedback/question. Once the server receives answers back from all the clients, it does corresponding analysis of the data and produce appropriate summarized representation as specified by the faculty. The server also maintains the score of every student, which can be used later to calculate student's grade.

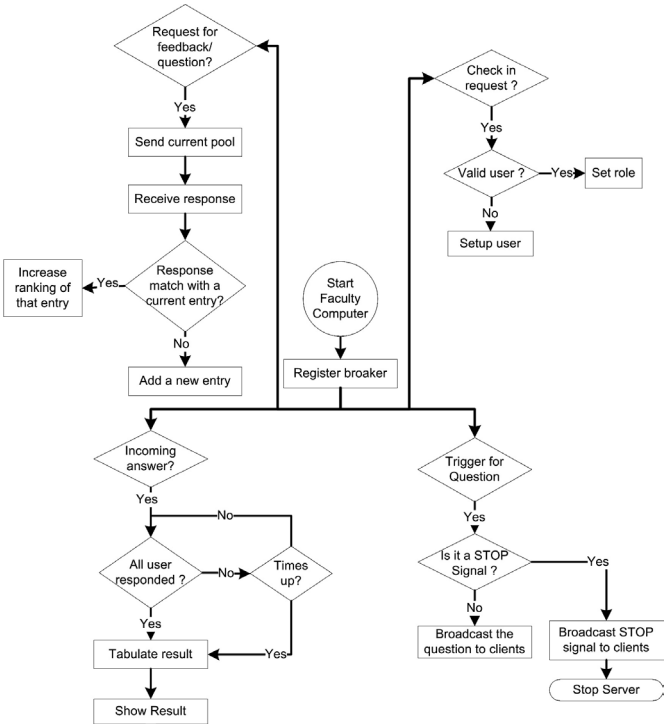

Figure 1. Server lifecycle.

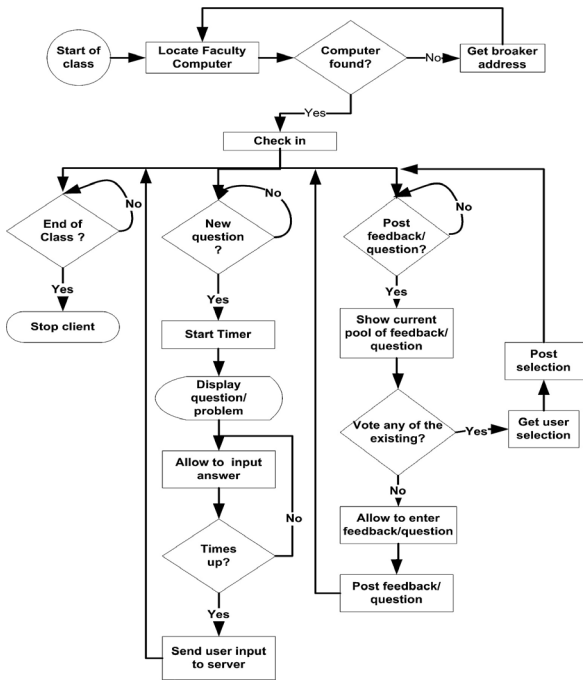

Figure 2. Client lifecycle.

B. Clients

Figure 2 shows the life cycle of the client. Once a student checks in, the app shows a standard screen where students can only submit new feedback/question or vote on one. If the app receives a new question from the server, the corresponding

activity that will render the given question into interactive entity will be executed. Every problem has a set amount of time to answer (which the faculty can assign) and a visible timer starts counting down to allow the students to see how much time they have left to answer the question. Once the answer of the student is received (or the timer runs out) and sent to the server, the app goes back to its root screen. At any time during the class, students can initiate a session to post a feedback/question or vote on an existing one anonymously.

IV. System Development

The current implementation of the server is in Java and the client is on Android. The server's components are:

- User interface: To manage and monitor the system.
- Encoder and Decoder: To store, transmit and convert the question and answer in the server and to client applications. A lightweight XML format is developed for encoding.
- Result processing: Data processing capabilities built into this module to process answers back from the clients and to produce proper representation.
- Blackboard integration: To import student information and export student score to course management system.
- Trigger management: To synchronize question broadcasting and answer propagation and timing management.
- Meta-language: A Meta language is developed to describe the questions and to render them in a general way across different platforms. It is XML based and has similarity with the encoding scheme mentioned above.
- Question editor: To easily add, delete and edit questions.
- User management: To manage user roles and information.

The client app has platform specific components and also the following:

- User Interface: To allow students to interact with the given problem and to initiate/vote anonymous questions/feedback.
- Networking module: This module is responsible for transmission and reception of questions, their answers and student feedback/questions and any other server messages. To properly communicate with the server, an asynchronous communication and synchronization protocol is developed to satisfy timely execution of problems and propagation of answers back to the server.
- Render module: This module render the encoded question sent from the server in the client device. This module determines which Android activity should be initiated for a question, initialize the activity with proper parameters as sent from the server and keep in consideration the target screen dimension, resolution and orientation for rendering a question.
- Log-in module: This component is responsible for discovering the server, checking user information and setting appropriate role of the user. This module allows the client to work per session basis and will automatically logs out, if it receives a stop signal from the server.

A. Extensibility

Building each possible problem type within the client is impossible and not practical. This will not only make the client overly bulky to run in mobile platform, but will also make it domain centric. To overcome this, the activity (or the android app) that facilitates interactive problems is completely separated from the application logic of the MRS client software. Figure 3 shows the life cycle of such interactive problem app. This approach makes it possible to integrate and execute any interactive problems developed by third party developers into the MRS software. The renderer module of MRS software can run any android activity (and app) as long as following conditions are met:

- Name of the package for the target app should be same as the question type in server.
- The incoming question will be delivered to the app through an Android Intent using the encoding scheme. Therefore the developer should use the decoding method in the project library to work with the given question.
- Answers from the interactive app should be returned to MRS using an Android Intent and using the encoding method in the project library.

Constrained with above conditions, anyone can develop their own interactive problem apps in any domain and use the MRS software to incorporate interactive problem solving in class. Third party developer can also decide on the level of interactivity they want to incorporate in their apps and the way to tackle Android's activity life cycle for their app.

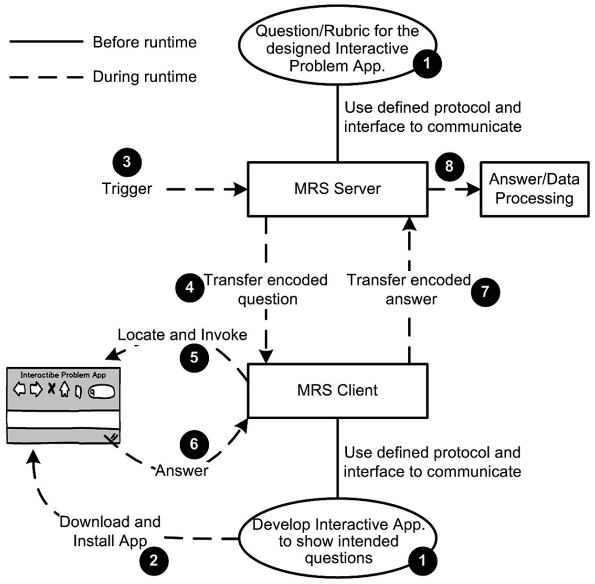

Figure 3. Lifecycle of interactive problem app.

B. Communication

A prefix-based communication scheme is devised for all messages, where different parts of the message are separated by a dedicated delimiter symbol. Any acknowledgement is piggy-backed with an outgoing message to minimize transmission overhead. The prefixes distinguish each message and the server or the client process an incoming messages according to the prefix. The size of the prefixes and the delimiter symbol are kept small in order to reduce the transmission overhead. Underneath the custom transport layer protocol, UDP is used to transfer packets between clients and

server. Since each of the messages occupied single packets, there is no need to sequence them, saving us the overhead for sequencing and blocking communication.

C. Software Matrices

To examine the quality and maintainability of the code, we use a static code analysis tool named STAN [10]. STAN analyzes the structure of the code and visualizes the design, to measure quality of the code and to report design flaws. STAN supports a set of selected metrics, suitable to cover the most important aspects of structural quality. Below we present two such measures that show the overall quality of the code.

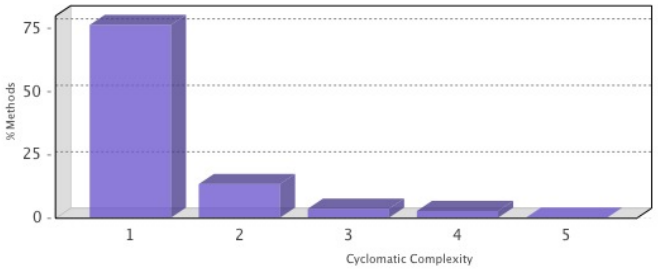

Figure 4. McCabe Cyclomatic Complexity.

Figure 4 shows McCabe Cyclomatic Complexity [11] of the code. We used this measure as it can be accurately calculated from the static call graph and determines code's structural complexity. Studies [12]-[13] show a correlation between a program's Cyclomatic Complexity, where code with higher Cyclomatic Complexity have higher probability of inducing errors during later maintenance. In that regard, it is evident from Figure 4 that the Cyclomatic Complexity values for MRS falls within the low risk and simple program range [14] and therefore the software is easy to maintain and extend. We also evaluated the code using object-oriented metrics found in [15]. Table I shows the values calculated by STAN for the developed code. The value for each of the matrices falls within desired ranges, which justify our claims regarding extendibility, manageability and maintainability.

TABLE I. QUALITY MATRICES.

Chidamber & Kemerer Metrics	Average
Weighted Method Per Class (WMC)	9.2
Depth of Inheritance Tree (DIT)	2
Number of Children (NOC)	1
Coupling between Objects (COB)	4
Response for a Class (RFC)	6.87
Lack of Cohesion in Methods (LCOM)	7.93

V. MRS AT WORK

One of the goals of the system is to provide users with easy to use interfaces and seamless user experience. In that regard, all user interfaces were made intuitive and easy to use. Before the start of the class, faculty has to setup in-class interaction (importing student info, questions, their answers/ rubrics and setup type of analysis, kind of report etc.) in the faculty computer using a graphical user interface. Once the interaction for the class is setup, students logged into the system and the class is in progress; the faculty can use a trigger (mouse click or a designated key in the keyboard) to broadcast a question to students. On the other hand, when a student initiates a check-in to the system, the clients first locates the faculty machine (MRS server) and once a session is established, credentials are validated so that the client can start accepting questions. After a successful login, the client shows the root screen, where the students can post anonymous questions/feedback to the faculty. Once a question is received the client invokes the corresponding activity to render the given question as described in Section IV.A. As mentioned earlier, each question has multiple interactive screens, which students can traverse back and fourth. Once the faculty machine receives answers back from all the clients, it does corresponding analysis of the data and displays it accordingly. Currently, several interactive problems are being developed with a planned deployment in the class of Fall, 2014. Currently MRS is operational and it's scalability, responsiveness and reliability parameters have been tested and they all fell within accepted ranges.

VI. CONCLUSIONS

In this paper, we present the design and development of MRS software that is targeted towards supporting classroom interaction using mobile devices. More specifically the goal is to facilitate interactive problem solving, submission, grading polling etc. by using the software. The architecture is presented and elaborated to exemplify the communication, maintainability and extendibility aspects. The result acquired from software quality data and the user interface shows that the system is easy-to-use, extendible and maintainable.

REFERENCES

[1] Mockus, L. and Edel-Malizia, S., "The Impact of Mobile Access on Motivation: Distance Education Student Perceptions", 17th Annual Sloan Consortium International Conference on Online Learning, 2011.

[2] Jones, A., & Issroff, K., "Motivation and mobile devices: exploring the role of appropriation and coping strategies", Research in Learning Technology, Vol. 5, No. 3, 2007.

[3] Roberts, J., Harvesting fragments of time. Mobile learning pilot project. Technical report, McGraw-Hill, 2003.

[4] Young, J. Mobile College App: Turning iPhones Into 'Super-Clickers' for Classroom Feedback, Chronicle of higher education, 2008.

[5] Top Hat Monocle, 2013, https://www.tophatmonocle.com.

[6] E-Clicker, Apple App Store, 2013.

[7] Anderson R., et. al., "Classroom Presenter: Enhancing Interactive Education with Digital Ink", Computer, Vol. 40, No. 9, pp. 56-61, 2007.

[8] Griswold, W., Simon, B., "Ubiquitous presenter: fast, scalable active learning for the whole classroom", ITiCSE, pp. 358, 2006.

[9] Berque, D. "An evaluation of a broad deployment of DyKnow software to support note taking and interaction using pen-based computers", Journal of CSC, Vol.21, No.6, pp. 204-216, 2006.

[10] Structure Analysis for Java (STAN), http://stan4j.com, 2014.

[11] McCabe, T., A Software Complexity Measure, IEEE Transactions on Software Engineering, Vol. 2, pp 308-320, 1976.

[12] Watson, A. and McCabe, T. Structured Testing: A testing methodology using cyclomatic complexity metric, NIST special publication, 1996.

[13] Clark, M., "Measuring Software Complexity to Target Risky Modules in Autonomous Vehicle. Systems." AUVSI North America Conference, 2008.

[14] C4 Software Technology Reference Guide, Software Engineering Institute, Carnegie Mellon University, 1997.

[15] Chidamber, S.R., Kemerer, C.F., A Metrics Suite for Object-Oriented Design. IEEE Transactions of Software Engineering, pp. 476–493, 1994.