Developing electronic classroom response apps for a wide variety of mobile devices – Lessons learned from the PINGO project

Michael Sievers
University of Paderborn
Computer Science Education Group
Fürstenallee 11, 33102 Paderborn, Germany
michael_sievers@web.de

Wolfgang Reinhardt
University of Paderborn
Computer Science Education Group
Fürstenallee 11, 33102 Paderborn, Germany
wolle@upb.de

Dennis Kundisch
University of Paderborn
Information Management & E-Finance
Warburger Str. 100, 33098 Paderborn, Germany
dennis.kundisch@wiwi.upb.de

Philipp Herrmann
University of Paderborn
Information Management & E-Finance
Warburger Str. 100, 33098 Paderborn, Germany
philipp.herrmann@wiwi.upb.de

ABSTRACT
Classroom response systems (CRSs) have proven to turn students into active participants in lectures. Thus, these systems help to improve the students' learning performance in traditional head-on lectures. Peer Instruction is a teaching and learning approach that makes very specific use of Classroom Response Systems elements: similar to the ask-the-audience lifeline in *Who Wants to Be a Millionaire?*, students get involved in the lecture by using clickers to answer multiple-choice questions posed by the instructor. Depending on the answer distribution the students are encouraged to discuss their answers with their peers. After some time of peer discussion the instructor poses the question again. In our efforts to develop a web-based application to support Peer Instruction in very large groups, we have faced several issues that we did not take into consideration in the initial design phase of the application. In this paper we share some lessons learned from the development process and report about some oddities found.

Author Keywords
classroom response systems, mobile development, lessons learned, development issues, platform independence, peer instruction

INTRODUCTION
Many researchers have concluded that the traditional head-on lecture-style courses contribute only little to the students' understanding of central concepts in their respective domains (e.g., Bleichner et al. 2000; Crouch et al. 2007). At the same time, it is commonly agreed upon that students will develop complex reasoning skills most efficiently when they actively participate in the subject matter (e.g., Crouch et al. 2001; Hake 1998). Thus, active participation fosters the students' learning process much better than traditional head-on lectures. So-called *Classroom Response Systems* (CRSs) are one way to motivate students in active participation in the lectures. A CRS enables instructors to pose questions in learning settings and supports them in gathering immediate feedback from the students. Researchers from various research domains have extensively documented the benefits of CRS (Fies and Marshall 2006; Roschelle et al. 2004).

Over the past years, many technology-enhanced CRSs have been developed that allow students to cast their votes using so-called *clickers*. The main advantages of such electronic CRSs over non-technical methods are the possible anonymity of responses (Draper and Brown 2004), the ability to immediately visualize response graphs for the class, as well as durability and extended means of data analysis. Since electronic CRSs also result in higher acceptance rates compared with conventional ones (Stowell and Nelson 2007), most instructors now rely on electronic systems in their classes (e.g., Cleary 2008).

One specific teaching approach that makes use of the prominent CRS elements is Peer Instruction (PI). PI (Mazur 1997a) is a cooperative teaching and learning approach that is well suited to involve students even in large auditoriums. PI is technically similar to the ask-the-audience lifeline in *Who Wants to Be a Millionaire?*; students get involved in the lecture by using (physical or electronic) clickers to answer multiple-choice questions posed by the instructor. The questions are designed to motivate students and to unveil potential difficulties with the course topics and material. If the questions are not answered correctly, the course participants are encouraged to discuss their answers with their close peers or the presentation is repeated in a modified way (depending on the percentage of wrong answers). The motivational and learning benefits of PI have been shown and reproduced in many empirical studies using varying methods (e.g., Crouch and Mazur 2001; Crouch et al. 2007; Fies and Marshall 2006). However, until today, PI has been rarely used in very large groups, i.e., groups of (far) more than 350 students. One major reason for the comparably slow

dissemination of PI is the expensive infrastructure for electronic CRS that is mostly driven by physical clickers and/or software licensing fees. When electronic clickers applications are used, they are often limited to one operating system, only work with a limited number of participating clients or are hard to use (Stuart et al., 2004; Herreid, 2006; Lasry, 2008; Salemi 2009; Kundisch et al., 2012). Moreover, there are complaints when students have to buy their own physical clickers (Patry, 2009). To design a technically and didactically favorable environment for the application of PI to very large audiences, we initiated the PINGO (Peer Instruction for very large Groups) project (Kundisch et al. 2012). The goal of the PINGO project is to develop an open, scalable, web-based and easy-to-use PI application for students and instructors.

In this paper we describe challenges in choosing the right platform for developing mobile applications that should be used by a large number of heterogeneous devices. Moreover, we report some oddities that we faced during the development of the web-based mobile application for PINGO. We close the paper with some conclusions that can be drawn from our experience.

CHALLENGES IN DEVELOPING MOBILE APPLICATIONS

When developing an application targeting mobile devices, the development process can be optimized regarding various parameters. In our specific scenario a very broad range of devices should be supported while minimizing the overall development effort. Three options were evaluated:

1. multiple native applications, one per device platform,
2. one multi-platform application and
3. a web-based application.

Building a native application for every supported device was no option at all, because of the huge development effort. The main reason is the diversity of platform specific programming languages, Application Programming Interfaces (APIs) and development tools. For example, source code and know-how from an iOS application written in Objective-C can hardly be transferred to an Android application written in Java.

In order to mitigate the differences between different platforms from the developers' point of view, there was the option to use a cross-platform-framework like Appcelerator Titanium[1] or PhoneGap[2]. Thereby a platform agnostic set of programming language and APIs is used, which is provided by the framework. In the end, there is conversion step, which creates platform specific applications based on only one code base. The drawback of this option is that the generalization of platform-specific APIs is only achieved only up to a certain point. For example, some well-known UI widgets are only available on one specific platform. If a developer wants to use such widgets, platform-specific code is needed. Appcelerator Titanium, for example, provides much platform-specific functionality at platform-specific namespaces. This enables a developer to make an application as native as possible but, at the same time, produces platform specific code, which had to be avoided with respect to the development effort.

Both options of developing an application for mobile devices have in common that they need some kind of a deployment infrastructure (e.g., the Apple App Store or Google Play). Such an infrastructure introduces issues, like delayed delivery of a new application version or rejection of an application by the infrastructure provider. Besides, supporting a broad range of mobile devices includes devices such as windows or Linux based laptops or tablets as well. However, these devices are supported by neither mobile cross-platform-frameworks nor the well-known deployment infrastructures.

Therefore, we decided to build a web-based application. From our point of view, this approach reaches the most devices and operating systems while minimizing development efforts. This approach is achievable because an increasing number of mobile devices integrate powerful web browsers, capable of technologies like JavaScript, CSS and HTML. Thus, it is rather unproblematic to run feature-rich web-based applications on these devices. Moreover, a web browser provides a platform agnostic runtime environment, eliminating the use of platform specific APIs. Also, no deployment infrastructure is needed. The user only has to enter a known URL into the device's browser or scan a given QR code.

ODDITIES OF WEB-BASED APPLICATIONS ON MOBILE DEVICES

When developing web-based applications, the browser determines the range of functions available to the application. That means that only technologies provided by all browsers on all devices that should be supported can be used by the application. Thus, the choice of which devices to support has a great influence on the usable technologies, because the 'weakest' device is decisive.

In the case of PINGO, the devices that determined the usable technologies were those running Android 1.6 or Symbian S60 3rd edition. Despite the widespread proliferation of latest smartphones, netbooks, and tablets, many of the mobile devices owned by university students still run these operating systems. Therefore, mobile web frameworks like jQuery

[1] http://www.appcelerator.com/

[2] http://phonegap.com/

mobile[3] or Dojo Mobile[4] could not be used, because the named devices lacked sufficient CSS support. Trying to circumvent missing CSS features with more mature web development approaches, such as tables and images, does not work in every case either. For example, replacing a CSS animated spinning wheel with an animated GIF image does not work on Android 1.6 because there is no support for animated GIF images.

Moreover, a web-based application cannot control the mobile device as much as native applications can do. For example, a change of the device's power state (e.g., to suspend or standby) cannot be interrupted by a web-based application. It cannot even be handled prior to its occurrence. The execution of the application will simply be stopped. This is problematic in case of timing-sensitive tasks, for example for periodic heartbeats sent by the client to indicate its presence. In our specific scenario, persistent Socket.IO[5] connections between the application and the backend collapsed due to the fact that the backend expected the client to be gone because it received no more heartbeats after the client device changed its power state. Even more problematically, after changing its power state back to active the device expects the persistent connection still to be fully functional. This results in client messages sent to the backend, which are silently being dropped because the backend has already closed the connection.

These cases can be identified by the developed application by using some sort of periodically timed, heuristic checks. However, these heuristics do not indicate a change of the device's power state in every case. A better alternative is to work without persistent connections at all and use stateless communication patterns, such as HTTP-REST.

As web-based applications cannot be tested with each and any targeted mobile device prior to deployment, the backend has to validate the consistency with respect to the agreed patterns for each message from the client. While developing PINGO, unchecked client messages broke the backend several times because some client devices did not behave as expected. When deploying a web-based application, one cannot trust the executing device to behave correctly. For example, there may occur some unexpected problems if the user disables JavaScript execution or prohibits client-side storage.

Incompatibilities between different mobile browsers have to be handled as well. This problem is not as crucial as on the desktop side, because of the dominance of WebKit-based mobile browsers. Nevertheless, a web-based application may not work as expected when started on a non-WebKit browser, such as Microsoft's mobile Internet Explorer (IE). For example, using the well-known JavaScript function console.log will silently break a web-based application on mobile IE because this function does not exist in mobile IE's JavaScript context.

CONCLUSION AND RESEARCH OPPORTUNITIES

Based on our work on developing a web-based application to support Peer Instruction in very large groups, in this paper we have discussed general challenges that have to be dealt with when developing mobile applications. We outlined that the development has to take into consideration the specifics of the devices to be supported and how they determine which application framework and technology may be used. We found that cross-platform-frameworks like PhoneGap or Appcelerator Titanium may be well suited to develop simple applications that merely present web-based content. If native functions of the targeted operating system shall be used, knowledge in the respective programming languages is needed. Thus, we have decided to develop the mobile application for the PINGO project using handcrafted HTML and CSS as it seemed to be the most cost- and time-efficient approach. During the development we found several oddities that we reported in the above section.

Our first prototypes are now used in practical evaluations in courses in business information systems (1000+ students enrolled), teaching and learning (500+ students) and computer science education (20+ students). So far, the students gave very favourable feedback for both the general idea of a low-cost, universally usable Peer Instruction application and the mobile application in particular.

The course in business information systems was an ideal candidate to evaluate the prototype as there were more than 1,000 enrolled students and 95% of these students possessed one or more web-enabled devices. Amongst others we used the Technology Acceptance Model (Davis et al. 1989) to evaluate the PINGO application. The average perceived ease of use among the 438 respondents was evaluated with 6.22 out of 7. The average perceived usefulness was evaluated with 4.93, the average attitude towards using with 5.49 and the average behavioral intention to use with 5.70 (see more detailed information on the evaluation in Kundisch et al. 2012; Reinhardt et al. 2012). All of these results are a strong positive indicator for the future usage of the system.

We will continue to improve the overall design of our application using the early feedback from the prospective users and make the PINGO application framework open source after maturing all parts of the infrastructure.

---

[3] http://jquerymobile.com/

[4] http://dojotoolkit.org/features/mobile

[5] http://socket.io/

Finally, we want to give some advice to developers of mobile learning applications that target a very large number of parallel users:

- always develop with the 'weakest' device in mind (hint: a web-based application that runs on Android 1.6 will run almost everywhere else as well),
- prefer stateless communication over persistent connections,
- test your application on as many different mobile devices as possible,
- when you rely on message-based communication between client and server, make sure that you check each client message, as you cannot trust the client.

ACKNOWLEDGMENTS
The research reported in this paper was supported by the prize of the award for innovation and quality in teaching at the University of Paderborn.

REFERENCES

Beichner, R.J., Saul, J. M., Allain, R. J., Deardorff, D. L., and Abbott, D. S. 2000. ''Introduction to SCALE-UP: Student-centered activities for large enrollment university physics,'' in *Proceedings of the 2000 Annual Meeting of the American Society for Engineering Education*, Session 2380.

Cleary, A. 2008. "Using Wireless Response Systems to Replicate Behavioral Research Findings in the Classroom," *Teaching of Psychology* (35), pp. 42-44.

Crouch, C.H., and Mazur, E. 2001. "Peer instruction: Ten years of experience and results, "*American Journal of Physics* (69), pp. 970-977.

Crouch, C.H., Watkins, J., Fagen, A. P., and Mazur, E. 2007. "Peer Instruction: Engaging Students One-on-One, All At Once," in *Reviews of Research-Based Reform Curricula in Introductory Physics*, E. F. Redish, and P. Cooney (eds.) Reviews in PER Vol. 1.

Davis, F.D., Bagozzi, R.P., Warshaw, P.R. 1989. "User acceptance of computer technology: A comparison of two theoretical models", *Management Science* 35(8), pp.982-1003.

Draper, S. W., and Brown, M. I. 2004. "Increasing interactivity in lectures using an electronic voting system," *Journal of Computer Assisted Learning* (20), pp. 81–94.

Fies, C., and Marshall, J. 2006. "Classroom Response Systems: A Review of the Literature," *Journal of Science Education and Technology* (15:1), pp. 101-109.

Hake, R. R. 1998. "Interactive engagement versus traditional methods: a six-thousand student survey of mechanics test data for introductory physics courses," *American Journal of Physics* (66), pp. 64-74.

Herreid, C. F. 2006. "'Clicker' Cases: Introducing Case Study Teaching Into Large Classrooms, " *Journal of College Science Teaching* (36:2), pp. 43-47.

Kundisch, D., Herrmann, P., Whittaker, M., Beutner, M., Magenheim, J., Reinhardt, W., Sievers, M., Zoyke, A. (2012). Designing a web-based application to support Peer Instruction for very large Groups. Submitted to *International Conference on Information Systems* 2012.

Lasry, N. 2008. "Clickers or Flashcards: Is There Really a Difference?," *The Physics Teacher* (46), pp. 242-244.

Mazur, E. 1997a. Peer Instruction: A User's Manual. Englewood Cliffs, NJ: Prentice-Hall.

Patry, M. 2009. "Clickers in Large Classes: From Student Perceptions Towards an Understanding of Best Practices", *International Journal for the Scholarship of Teaching and Learning* (3:2), pp. 1-11.

Reinhardt, W., Sievers, M., Magenheim, J., Kundisch, D., Herrmann, P., Beutner, M., Zoyke, A. 2012. "PINGO: Peer Instruction for very large groups," *Proceedings of the 7th European Conference on Technology Enhanced Learning*, pp. 507-512.

Roschelle, J., Penuel, W. R., and Abrahamson, L. 2004. "Classroom Response and Communication Systems: Research Review and Theory," *Annual Meeting of the American Educational Research Association*, San Diego, CA, 2004.

Salemi, M. K. 2009. "Clickenomics: Using a Classroom Response System to Increase Student Engagement in a Large-Enrollment Principles of Economics Course," *The Journal of Economic Education* (40:4), pp. 385-404.

Stowell, J. R., and Nelson, J. R. 2007. "Benefits of Electronic Audience Response Systems on Student Participation, Learning, and Emotion", *Teaching of psychology* (34:4), pp. 253-258.

Stuart, S. A. J., Brown, M. I., and Draper, S. W. 2004. "Using an electronic voting system in logic lectures: one practitioner's application", *Journal of Computer Assisted Learning* (20), pp. 95-102.