Informa: An Extensible Framework for Group Response Systems

Matthias Hauswirth

University of Lugano, 6904 Lugano, Switzerland
Matthias.Hauswirth@unisi.ch
http://www.inf.unisi.ch/faculty/hauswirth

Abstract. Classroom clickers, also called group response systems, represent a form of technology-enhanced learning. An instructor can pose a question to the class during a lecture, and students can use their clicker devices to submit their answers. The system immediately aggregates the submissions and presents feedback to the instructor (and possibly the class).

This paper describes Informa, an extensible framework for building software-based group response systems. Informa is implemented as a distributed Java RMI application and distinguishes itself from traditional clickers in two key aspects: First, it allows for plug-ins to define the kinds of problems that can be posted (beyond the common multiple-choice). Second, it provides several levels of session anonymity, from completely anonymous sessions where the teacher does not know which student submitted which answer, all the way to authenticated sessions where students need to login when they join.

We have evaluated Informa in a pilot study during an undergraduate programming course, and we have found it to greatly enhance our insight into the students' understanding of the material.

Keywords: technology-enhanced learning, classroom response systems.

1 Introduction

Lecturing is probably the most often used teaching method in higher education. However, lecturing is not easy, and many lectures are evaluated as largely ineffective by students [1]. In his seminal book [2], Penner states that the problem is not the lecturing method, but its poor execution. In particular, he emphasizes the importance of continuous feedback from students to the lecturer. He goes as far as declaring invalid the methodology of a study, where (for the purpose of repeatability) taped lectures were played back to students, thereby completely inhibiting any feedback ("Blind flight" scenario in **Figure 1**).

The effectiveness of a lecture greatly depends on the education (in the subject matter as well as in pedagogy) and the personality of the teacher. Teachers often solicit feedback from students by asking questions ("Question" in Figure 1). However, even a well educated teacher with a well-suited personality is limited in

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 271–286, 2009.

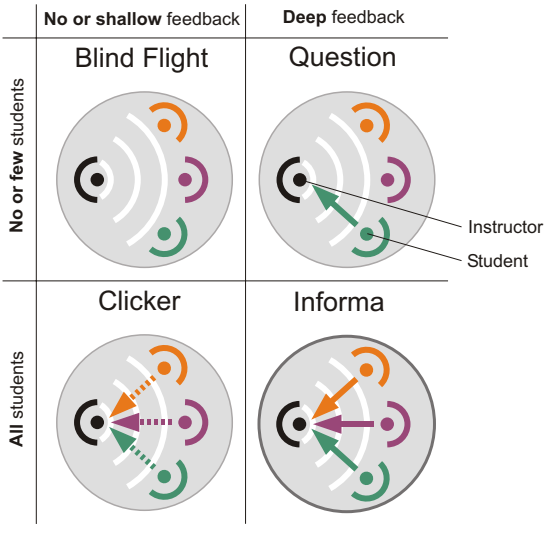

Fig. 1. Four Scenarios for Feedback in a Classroom

getting representative feedback. The problem in continuously evaluating how well the students understand the presented material is twofold. (1) With a large class size, there generally is not enough time to continuously assess every student's understanding, and (2) some students are reluctant to let the instructor or fellow students know that they have problems following the presentation.

Recent innovations in educational technology address these two issues. Group response systems [3,4], also called "classroom clickers", are a technological intervention for the continuous gathering and evaluation of feedback about student learning ("Clicker" in Figure 1). Clickers are little remote controls distributed to students before class. During the lecture, students are repeatedly asked multiple-choice questions which they answer using their clickers. The results are immediately tabulated and can be presented to the class, usually anonymized in the form of histograms.

The pedagogical motivations for using clickers are manifold: Clickers allow a more interactive teaching style even with very large class sizes; they allow for anonymous and immediate feedback from students; they increase class participation of shy students; they allow (or force) students to submit an answer even if they are not sure about a point; and they allow the instructor to regularly measure the standing of all students, not just the good ones.

Traditional clicker devices are special purpose remote controls with a limited user interface consisting of only a small number of buttons (corresponding to the maximum number of choices in a multiple-choice question). Some educators have used laptops instead of clicker devices [5], an approach that can easily be adopted at institutions where all students bring laptops to class. Other researchers have used programmable calculators [6], and one can envision other

ubiquitous devices being used as clickers, such as mobile phones or PDAs. On devices that can execute arbitrary applications, a clicker can be implemented as a software application, and can thus provide capabilities that go far beyond the classical clicker device.

However, to our knowledge, the potential of such software-based clickers has not been fully exploited so far: While hardware clickers are limited to multiple-choice questions, software clickers allow students to submit much richer information ("Informa" in Figure 1). Moreover, software clickers enable several degrees of anonymity, from completely anonymous sessions where the teacher does not know which student submitted which answer, all the way to authenticated sessions where students need to login to join.

In this paper we introduce a system we call Informa (Integrated Formative Assessment). Informa enables the most desirable of the four scenarios shown in Figure 1: A classroom where the instructor gets *deep feedback* from *all students*. In Section 2 we outline our design goals. Section 3 describes Informa, and Section 4 presents a usage example. In Section 5 we define the different types of anonymity useful for a classroom response system, and in Section 6 we discuss Informa's extensibility through problem-type plug-ins. Section 7 presents the results of a pilot study using Informa, Section 8 discusses related work, and Section 9 concludes.

2 Goals

The primary goal of this work is the development of an effective pedagogical approach. The purpose of the infrastructure we present in this paper is to fulfill that goal. We thus do not strive to provide a framework that drives entire lectures, and we avoid the known problem of *over-scripting* [7], but we aim at a blended learning approach using technological support only where needed.

Given this premise, it is essential that our infrastructure has a low cost in terms of deployment and that it is easy to use. Students and teachers should not have to spend excessive amounts of time installing and maintaining the collaborative learning infrastructure. It should be possible for students to quickly install and run the student software during the first lecture of a course. Instructors should be able to install the software by simply downloading and running a program, without the need for any configuration or complicated server setup. The software, both the student and the instructor applications, should work on any platform commonly used by teachers or students.

Moreover, the software should gracefully handle latecomers, allowing students to connect or reconnect to a session at any time. This aspect is important in two ways: Besides addressing the issue of students who arrive late for classes, this also overcomes problems with students who terminate their application (e.g. because of a system crash, or because they accidentially close the application).

We strive for an infrastructure that is extensible along several axes: (1) teachers should be able to choose an anonymity level for a given session, (2) teachers should be able to compose problem sets for their lectures and create new

problems, and (3) developers should be able to develop plug-ins to support new types of problems (e.g. beyond simple multiple-choice questions).

3 Informa

In this section we describe the Informa framework including its architecture, its communication protocol, and the pedagogical script behind its use.

3.1 Architecture

Figure 2 shows the system architecture of Informa. Informa consists of two applications. The *student* application, which students run on their laptops, represents the actual "clicker" and allows students to solve the posted problems and to submit their solutions. The *instructor* application runs on the instructor's computer and maintains the database of problems, manages sessions, and aggregates students' answers. Moreover, it provides a user interface for the instructor to manage and post problems, and to visualize the aggregated student solutions.

The student applications communicate with the instructor application over the (possibly wireless) network. We have implemented Informa as a distributed system using Java RMI. Using Java allows us to run on most operating systems installed on student laptops. Moreover, unlike a web-based application, a rich Java application enables the use of an extensive collection of open source libraries for implementing rich direct-manipluation GUIs for editing and solving problems.

3.2 Pedagogical Script

A teacher uses Informa in two contexts: (1) before class to prepare a session, and (2) in the classroom during a lecture to run a session. In both cases the teacher runs the Informa instructor application. The functionality available in both contexts is the same: a teacher can compose problem sets and create new problems during preparation as well as during class.

In the classroom, the teacher starts an Informa session at the beginning of class. An Informa session consists of one or more steps. Each step represents the execution of the following pedagogical script in which students are to solve a

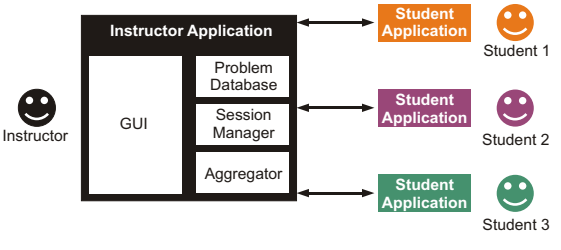

Fig. 2. System Architecture

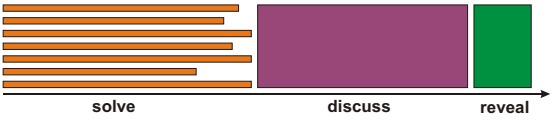

Fig. 3. One Step in an Informa Session

given problem. A step is partitioned into the three phases shown in **Figure 3**: First, in the "solve" phase, the teacher puts up a problem for the students to solve at their computers. Students work individually on their solutions and submit them when they are done. Second, after all submissions are in, or when the teacher decides the time is up, the teacher displays an aggregate visualization of all submitted solutions. In this "discuss" phase she moderates a discussion with the goal of identifying and explaining the good and bad solutions. Finally, in the "reveal" phase, the teacher reveals the correct solution and explains it to the class.

As Figure 3 shows, the students work independently during the "solve" phase. We intentionally designed our approach this way to encourage each student to think deeply about the given problem. The collaborative aspect of our approach manifests itself in the "discussion" phase. After each student has spent some time working on the problem, they now see how their individual solution relates to the overall view of the class. Note that at this point the correct solution is not revealed as yet, so the class is left on their own to collaboratively determine the correct solution. It is this phase that triggers the pedagogically most valuable discussions. After this discussion, the teacher may "reveal" the correct solution (or an example of one of many correct solutions), and explain any remaining issues.

Note that Informa does not require that a problem has correct solutions. Thus, a teacher may run a step to poll the students about their opinions on a given issue, or to gather their subjective judgements.

3.3 Protocol

In Informa, the instructor application is responsible for maintaining the session state. It is usually running during the entire duration of a lecture. Students are able to join or leave a session at any time, but they usually join at the beginning and leave in the end.

Figure 4 shows a high-level view of the RMI-based protocol between the instructor and student applications. It shows the following four scenarios:

Student joins session. Student applications join a session by contacting the instructor application. They find the instructor (an RMI remote object) by looking it up in an RMI registry at a well known port on the instructor's computer. The student application prompts the student for the IP address or hostname of the instructor's computer (the instructor application displays this address on the classroom beamer).

Once student applications have a reference to the instructor RMI object, they try to join the currently active session (see "Student joins session" in

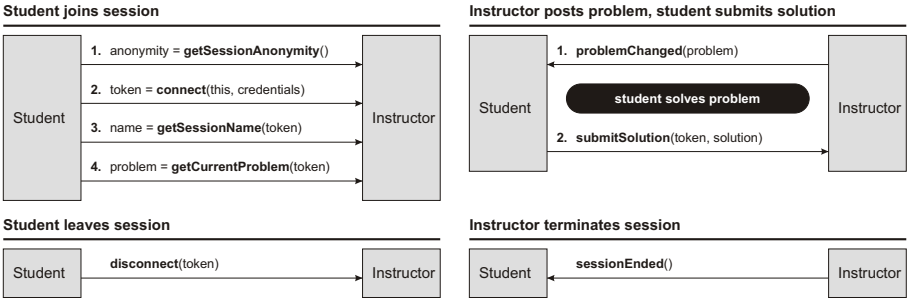

Fig. 4. RMI-based protocol between instructor and students

Figure 4). For this, they first enquire about the current session's anonymity approach (e.g. anonymous, or requiring login). Then they prompt the user for the necessary credentials depending on the given anonymity (e.g. user name and password for a session requiring a login) and connect to the session using these credentials. If the connection request is successful (e.g. user name and password are valid), the instructor returns a token that uniquely identifies the student and allows the student to participate in the session (all further methods of the instructor require a valid token). The student can then request the name of the current session (and other information), and retreive the currently active problem.

Using this approach, latecomers will retrieve the currently active problem, but will not have access to the previously posted problems. In general, the history of past problems is not retained in the student applications, since the intention of Informa is to drive a face-to-face classroom situation. However, the instructor application keeps a history of past problems and submitted solutions. In the future we might make this history available for students to review after class.

Instructor posts problem, student submits a solution. An instructor using Informa can decide to post a problem at any time. This leads to the instructor application notifying all students (`student.problemChanged()`) about the new problem. The student applications then present the new problem to the students, and the students spend some time solving the problem. Once a student is done, he can submit his solution, leading to a call of `instructor.submitSolution()`.

Student leaves session. A student may leave the session at any time. The instructor maintains a list of all students in the session in order to post new problems to all students. If a student explicitly disconnects (by calling `disconnect`), or if a student is unreachable when the instructor posts a problem, the instructor removes that student from the list.

Instructor terminates session. Finally, at the end of the lecture, the instructor terminates the session, notifying all the students with a call to `sessionEnded`.

Notice that both the student *and* the instructor application initiate remote method calls. This is necessary to allow the instructor to post new problems to the students without the students polling for updates. It also allows the instructor to notify students when it wants to terminate the session.

4 Example Usage Scenario

This section introduces the Informa system and the related teaching methodology with an example: the preparation and teaching of a lecture in a Java programming course.

4.1 Preparation

The instructor prepares for her lecture by building a set of problems she intends to post during class. She uses the instructor application to prepare her problem set. This tool allows her to manage existing problems in her database and to interactively create new problems of any supported type. She can also import problems from existing problem databases, for example from a similar course she taught before.

Figure 5 shows Informa with a list of problems related to Java programming. The selected problem, a multiple-choice question, is previewed below the list. The instructor decides to modify this problem and Informa invokes the problem editor specific to multiple-choice questions shown in **Figure 6**. In addition to changing this problem, the instructor creates a few additional problems of various types before finishing her preparation by saving her problem database.

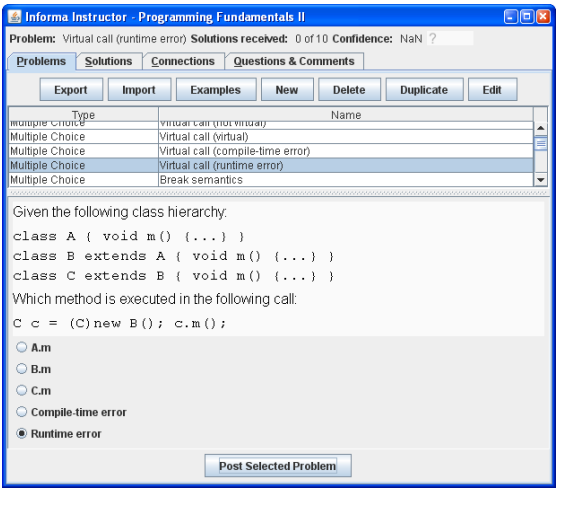

Fig. 5. Managing and Posting Problems

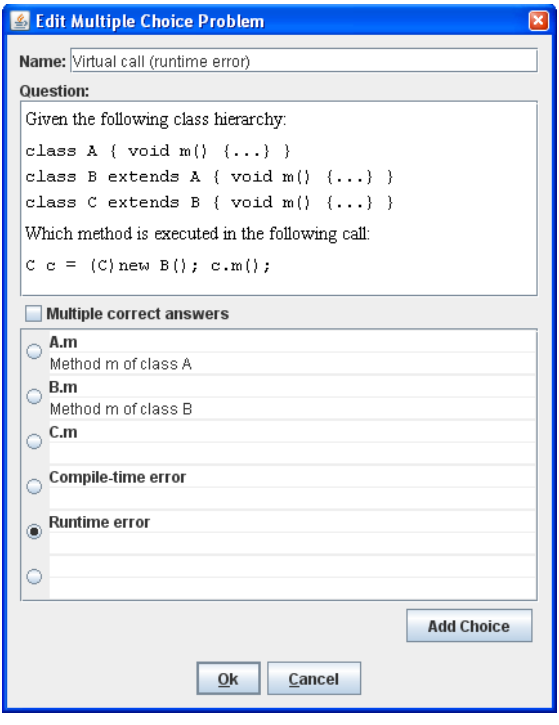

Fig. 6. Editing a Multiple Choice Problem

4.2 Classroom

At the beginning of the class the instructor starts an Informa session on her computer. She opts for an anonymous session, which means that she will not know which student submitted which solution.

After starting the session, the instructor application creates two different windows. The first window is the same she used during her preparations (Figure 5): it allows her to browse and manage her problem database. The second window presents information that she wants to communicate to the students. She configures her computer to extend her desktop across her monitor and the classroom beamer. This allows Informa to present the first window on her monitor while it projects the second window on the beamer, visible for all students.

The beamer window, shown in **Figure 7**, initially shows a welcome message for the session. This message includes information on how to download and start the student application, and information for how to connect to the server (i.e. the IP address of the instructor's computer).

The students start the Informa student application on their computers and connect to the server by entering the connection information presented on the beamer. Since the session is anonymous, they don't have to provide any additional login information.

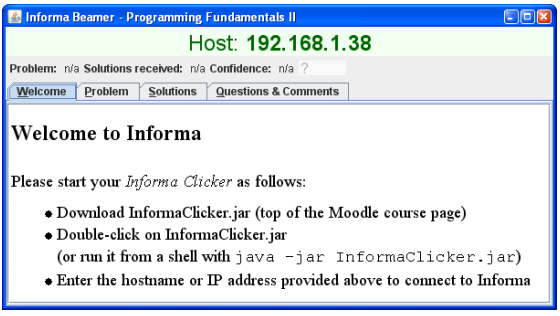

Fig. 7. Welcome Message on Beamer

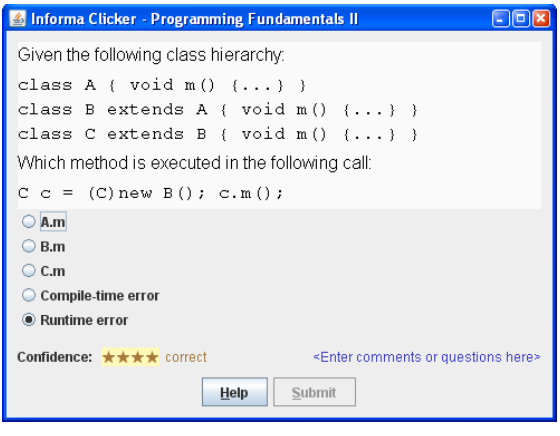

Fig. 8. Student Solving a Multiple-Choice Problem

Now the instructor picks and posts a problem to the class. All students will immediately see that problem on their screens, and they will be prompted to solve it (**Figure 8**). Since the instructor posted a multiple-choice question, the students just need to select the correct answer and submit.

The instructor can see the submitted solutions on her monitor, and she can see the number of outstanding solutions. She can close the problem before receiving all submissions, or she can wait until all students have submitted their solution.

We consider the next step the most valuable aspect of this teaching approach: The instructor application can project a visualization that summarizes all submitted solutions to the classroom beamer. **Figure 9** shows this visualization for the multiple-choice question she posted. It shows a histogram with the number of students who picked each of the choices. The instructor uses this histogram to prompt a discussion in class. Since the visualization does not reveal the correct answer, students can be asked to defend their choices or to explain whether and why an unpopular choice is correct. At the end of this discussion the instructor may reveal the correct answer(s) and provide clarifications and deeper explanations where necessary.

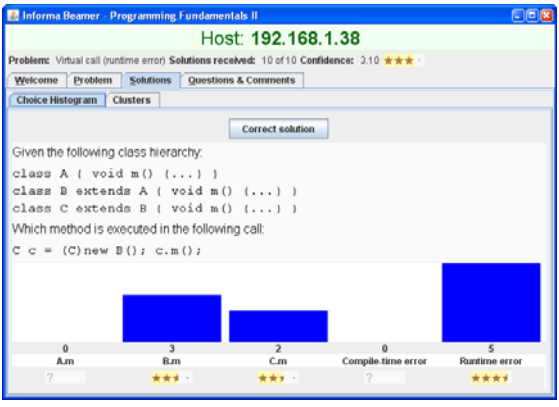

Fig. 9. Aggregate View of Multiple-Choice Solutions

5 Anonymity and Grouping

A classroom response system like Informa is ultimately confronted with the issue of anonymity: the question of whether it is possible to determine which student submitted a (possibly incorrect) solution. We have identified different approaches to anonymity:

Anonymous. Students are entirely anonymous.

User alias. Students pick an arbitrary alias when they connect. This alias is used whenever information about individual solutions is shown on the beamer, allowing each student to identify his own information.

User name. Students log in with a user name assigned to them by the instructor. This allows all students and the instructor to identify information about each individual student.

User name & password. Students connect with a user name and password. This type of session allows instructors to use Informa to keep class attendance information, or to conduct graded quizzes.

Group alias. Students enter a group alias of their choice. Unlike with a user alias, with a group alias multiple students are expected to enter the same alias (the alias of their group). While a user alias allows students to compete within a group of friends, a group alias allows students to compete between groups (e.g. the "skiers" vs. "snowboarders"). The specific group membership is irrelevant from the point of view of the instructor. The only purpose of forming groups is to increase motivation by fostering competition.

Group name. Students enter a group name assigned to them by the instructor.

Group pick. Students pick a group from a set of known groups.

The main goal of Informa is to enable *all* students to participate. Students are more inclined to answer a question if they feel comfortable making a mistake. Some students, even in the friendliest classroom, are reluctant to offer a

solution in which they lack confidence. The anonymity provided by the anonymous, user alias, and group alias approaches eliminates this barrier. The group name and group pick approaches, with small group sizes, lead to a certain loss of anonymity, but have the benefit of providing more information (e.g. the "Pascal programmers" group understands information hiding better than the "C programmers" group). On the other end of the spectrum, the two user-name-based approaches, user name and user name & password, have the advantage of automatically tracking specific students' progress. This allows the early detection of challenged students, enabling instructors to help these students before it is too late.

The aspect of anonymity is related to the aspect of competition. If students are completely anonymous there is little means for competition. The more information about a student is known, the more competitive the session becomes. Except for anonymous, all of the above approaches foster competition. Moreover, in group-based appoaches (group alias, group name, and group pick), submissions are aggregated and visualized by group instead of over the entire class, exposing the specific performance of each group. This enables competition *between* groups, providing a motivating setting without exposing indiviual students directly.

With Informa, the instructor decides on an anonymity approach at the start of the session. Depending on the approach, students connecting to the session will then be prompted for the required information.

6 Problem Types

Traditional classroom clickers focus on multiple-choice questions: the instructor offers a set of predetermined answers, of which the student has to choose the correct subset. In Informa, multiple-choice questions represent just one supported problem type. Informa is an extensible framework, where problem types are defined in plug-ins. In this section we describe two problem types in more detail, the standard multiple-choice questions and our new text highlighting problems.

Each problem-type plug-in provides editors for creating new problems (for the instructor), GUIs for solving a problem (for the students), and visualizations that aggregate the submitted solutions (for showing on the beamer). Developers can easily develop new plug-ins for new types of questions.

6.1 Multiple-Choice Questions

Section 4 shows the support for multiple-choice questions available in Informa. Figures 8 and 9 show a student's view of the problem and the aggregate view of all submitted solutions in the form of a histogram. Informa allows questions with an arbitrary number of choices. A given question can be configured as either a single or a multiple answer question. When designing the question, the instructor also indicates which choices are correct. Informa uses this information to highlight the correct answers in the histogram aggregating the submitted solutions. Since giving away the correct solution may not be the pedagogically

most effective approach, the highlighting of correct answers is initially disabled. The instructor can use the histogram to prompt a discussion with the students and highlight the correct answers only at the end of that discussion.

6.2 Text Highlighting Problems

This problem type consists of a text and a question that asks the student to highlight certain parts of that text. The example problem shown in **Figure 10** asks the students to highlight the name of each instance variable that has a reference type. The student has already highlighted `dateOfBirth`, is currently adding `mother`, and has not (yet) identified `name`.

While a multiple-choice problem has a fixed number of incorrect answers (distractors), text highlighting is a more open type of problem: a student is free to highlight any area(s) of the text he likes. Because it provides the student with less support (no choices to pick from), it may help uncover issues of understanding that would not have been detected with multiple-choice problems. Since the number of possible answers is large, the aggregation approach used for multiple-choice questions (i.e. histograms) becomes impractical.

For this reason we have developed the aggregation visualization shown in **Figure 11**. This visualization superimposes all solutions, leading to highlights with different intensity. The intensity of the highlights represents the number of solutions that highlighted the corresponding text. Intense highlights thus correspond to the majority solution, while divergences by small numbers of students (often corresponding to errors) show up either as faint highlights or as highlights of less-than-full intensity. The instructor can take action if the intensely highlighted text segments do not correspond to her expectations. In the case of minor issues like in Figure 11, the instructor may decide to explain that `String`

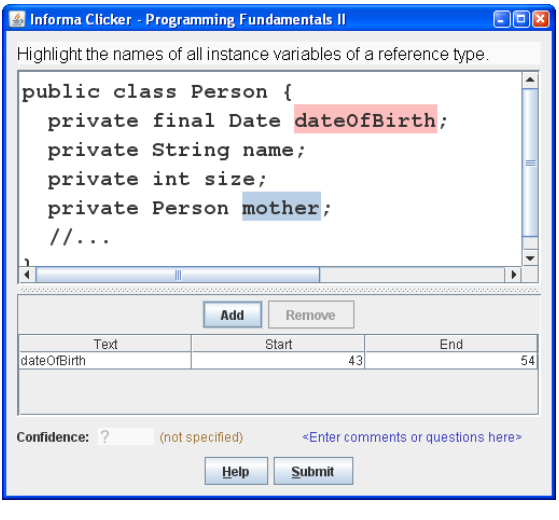

Fig. 10. Student Solving a Text Highlighting Problem

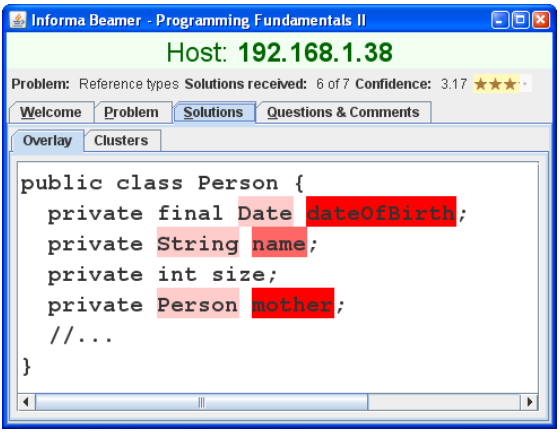

Fig. 11. Aggregate View of Text Highlighting Solutions

is a reference type, and she may comment that students were expected to select the name of the variable (not its type).

6.3 Writing New Problem Type Plug-Ins

To create a new problem type, a developer first has to decide how to store the information defining a problem and a solution by writing implementations of Informa's `Problem` and `Solution` interfaces. These classes are usually relatively lightweight. For example for the multiple choice problem type, the Problem class stores the question text and the list of possible choices. The Solution class stores the index (or indices) of the selected choices.

The Problem and the Solution class have to be serializable. This allows Informa to pass problems from the instructor to the student, and to return solutions from the student to the instructor. Moreover, it enables Informa to store problems in a file (the problem database is essentially a file containing serialized `Problem` objects).

```
public interface ProblemType {
  String getName();
  ProblemEditor createProblemEditor();
  SolutionEditor createSolutionEditor();
  SolutionAnalysisView[] createSolutionAnalysisViews();
}
```

Fig. 12. The ProblemType interface

Besides implementing these two model classes, supporting a new problem type also requires the developer to write an implementation of the `ProblemType` interface (see Figure 12), and to develop the necessary `ProblemEditor` and `SolutionEditor` GUI components. Moreover, the developer needs to develop

one or more `SolutionAnalysisView` to visualize the aggregate information over all submitted solutions (analog to the histogram for multiple choice solutions).

7 Evaluation

We evaluated Informa in a Java programming course (Programming Fundamentals II) at the University of Lugano (USI). All informatics students at USI bring laptops to class, and all our classrooms are equipped with beamers and wireless networks. At USI we have a high instructor to student ratio, allowing for small class sizes. While the main benefit of group response systems is to improve teaching with larger classes, we have found that even in our class of thirteen undergraduate students the use of Informa can be beneficial.

At the end of that course, we asked students to provide anonymous feedback on the use of Informa. Eleven students responded to our request. Most students saw benefits in using Informa, providing comments like:

> My answers were almost all wrong. After that I've started to read and prepare myself more.

> After answering the questions we would get the right answer and that sticks better to your mind.

Students also provided significant feedback for improving Informa. They ranged from bug reports (e.g. student applications that dropped connections) over feature requests (e.g. way to review problems and solutions after class) to pedagogical issues (e.g. difficulty of the problems).

The most important limitation the students identified was an issue with our pedagogical script: at the end of the first phase ("solve") in each step, the faster students have to wait for the slower ones. This serialization issue also was of great concern to the instructor. We plan to address this issue by allowing the instructor to post a batch of related questions at once. Another approach would be to provide students who have submitted the answer to their problem with extra reading material or bonus problems.

A practical issue that surfaced early on in the semester was that each latecomer had to ask the instructor for connection information (the IP address of the instructor's computer). We solved this problem by always displaying connection information on the beamer. Another solution would involve the use of a service discovery protocol to allow student applications to automatically find instructor applications.

We found Informa's extensibility to be really useful. We developed several specific problem-types for our course (e.g. for matching regular expressions, or for identifying the type of a Java expression). However, writing a new problem-type plug-in requires some effort. Over time we thus identified more general problem-type plug-ins, such as the text highlighting plug-in described in Section 6.2. We believe that, as we continue to use and extend Informa, we will end up with a collection of problem types that are broadly applicaple across course topics.

8 Related Work

Trees and Jackson [8] conducted a study involving 1500 students showing how the use of clickers can improve the effectiveness of large lectures. Our pilot experiment provides a first indication that our approach, which involves problem types beyond traditional multiple-choice, can also be beneficial in a lecture with a small number of students.

Roschelle et al. [9] survey clicker-related research and connect it to the broader educational literature. They argue that next generation clickers should focus on formative assessment (Informa's focus) and on effective means to visually aggregate student answers (which they do using overlaid plots of submitted polynomial equations [6], and we do e.g. with our text highlighting problem).

A more recent survey by Fies and Marshall [10] confirms these issues. It also points out the logistical difficulties to instructors and the cost of purchasing clickers for students. Our free, lightweight software solution mitigates these problems in situations where networked computers that run Java (desktops, laptops, PDAs, cell phones) are available in classrooms.

The "Classroom Learning Partner" (CLP) project at MIT uses pen-based Tablet-PCs to allow students to submit answers to questions. CLP proposes to aggregate student answers, but the CLP publications [11,12] only report on a tool that does not support aggregation. Our work differs from CLP in several ways. CLP has to correctly interpret the electronic ink before being able to reason about and compare answers. Because Informa's solution editors constrain the space of possible answers, the semantics of a solution are always unambiguous, and "freak solutions" (solutions that do not fit the context in which the problem was posed) are avoided. Moreover, CLP requires (expensive) special hardware and is based on proprietary software (Microsoft PowerPoint).

9 Conclusions

Informa is an extensible framework for group response systems. Using Informa in a classroom allows the instructor to get *deep feedback* from *all students*. Informa enables getting deep feedback about the understanding of students because of its pluggable problem types. Developers can create new plug-ins that require students to solve problems that go far beyond the selection of a choice in a multiple-choice question. Informa enables getting feedback from all students, because of its different student anonymity approaches which allow the instructor to pick a tradeoff between anonymity and competitiveness. We have successfully used Informa in our own courses. We plan to further improve the system and to release it as an open source product.

Acknowledgments. We would like to thank the anonymous reviewers for their insightful comments and the students at the University of Lugano for their feedback on improving Informa.

References

1. Bligh, D.A.: What's The Use of Lectures. Jossey-Bass (2000)
2. Penner, J.G.: Why many college teachers cannot lecture: How to avoid communication breakdown in the classroom. C.C. Thomas (1984)
3. Abrahamson, A.L.: An overview of teaching and learning research with classroom communication systems (CCSs). In: Proceedings of the International Conference of the Teaching of Mathematics (June 1998)
4. Duncan, D.: Clickers in the Classroom. Pearson Education, London (2005)
5. Draper, S.W., Cargill, J., Cutts, Q.: Electronically enhanced classroom interaction. Australian Journal of Educational Technology 18(1), 13–23 (2002)
6. Roschelle, J., Vahey, P., Tatar, D., Kaput, J., Hegedus, S.: Five key considerations for networking in a handheld-based mathematics classroom. In: Proceedings of the 27th Conference of the International Group for the Psychology of Mathematics Education (July 2003)
7. Dillenbourg, P.: Over-scripting CSCL: The risks of blending collaborative learning with instructional design, pp. 61–91. Open Universiteit Nederland, Heerlen (2002)
8. Trees, A.R., Jackson, M.H.: The learning environment in clicker classrooms: Student processes of learning and involvement in large courses using student response systems. Learning, Media and Technology 32(1), 21–40 (2007)
9. Roschelle, J., Penuel, W.R., Abrahamson, L.: Classroom response and communication systems: Research review and theory. In: Annual Meeting of the American Educational Research Association (April 2004)
10. Fies, C., Marshall, J.: Classroom response systems: A review of the literature 15(1), 101–109 (2006)
11. Koile, K., Singer, D.: Development of a tablet-pc-based system to increase instructor-student classroom interactions and student learning. In: Workshop on the Impact of Pen-based Technology on Education (April 2006)
12. Koile, K., Singer, D.: Improving learning in cs1 with tablet-pc-based in-class assessment. In: Second International Computing Education Research Workshop (submitted)