PABS - a Programming Assignment Feedback System

Lukas Iffländer[1] Alexander Dallmann[2] Philip-Daniel Beck[3] Marianus Ifland[4]

Abstract: Giving invididual feedback to students in large programming courses is time consuming and in most cases not feasible. In order to provide students with feedback we introduce PABS, a tool for automated feedback generation for programming assignments on the Java virtual machine. PABS is a web application giving students the opportunity to submit an arbitrary number of solutions and receive automatically generated feedback. It has been developed to scale with a growing user base by delegating the feedback generation to separate proccesses, called *agents*, that communicate asynchronously with the web application. At the University of Würzburg, PABS is successfully used in many courses that hand out programming assignments.

Keywords: E-Learning, automatic assessment, feedback

1 Introduction

Today methods from computer science are used in many different areas and therefore it is an increasingly popular field of study. Consequently, more students are trained in basic methods of computer science and programming. These students need to solve practical programming assignments as part of their education. Feedback can help students to stay motivated and assess their progress, but giving manual feedback is a very time consuming task. Especially in large courses with beginners it is not practical due to staff limitations. As a result it is desirable to automatically generate feedback whenever possible, so that time budgets can be moved to tasks that can't be automated like tutoring.

In this paper we describe a web-based tool for automatic feedback generation called PABS (German: ProgrammierAufgaben BewertungsSystem). It was created to support the practical programming course offered by the computer science faculty at the University of Würzburg, but has since then been extended to several other courses that are handing out programming assignments. PABS enables students to request and view automated feedback for their solutions, e.g. functional tests and style checks. For that purpose it uses Subversion repositories to store the students' code and provides a web-based user interface to display the code history as well as the generated feedback.

[1] University of Würzburg, Chair of Computer Science II, Am Hubland, 97074 Würzburg, lukas.ifflaender@uni-wuerzburg.de
[2] University of Würzburg, Chair of Computer Science VI, Am Hubland, 97074 Würzburg, alexander.dallmann@uni-wuerzburg.de
[3] University of Würzburg, Chair of Computer Science VI, Am Hubland, 97074 Würzburg, philip.beck@uni-wuerzburg.de
[4] University of Würzburg, Department of Computer Science, Am Hubland, 97074 Würzburg, marianus.ifland@uni-wuerzburg.de

In order to support a large userbase, the feedback generation in PABS is designed to be scalable. PABS uses *agents* to generate feedback. *Agents* run independently and use an asynchronous communication protocol to receive feedback requests and deliver generated feedback. *Agents* can be deployed to an arbitrary number of servers and dynamically started and stopped. This enables PABS to scale with the number of active users.

So far, PABS only supports programming assignments for languages based on the Java virtual machine (JVM) and has been exclusively used for Java programming assignments. Other languages based on the JVM have been tested, e.g. Scala and Groovy. In the future support for other programming languages will likely be implemented.

This paper starts with an overview of related work, describes the technical structure of the system in more detail and ends with a report of our experience with the practical application during the last years including statistics.

2 Related Work

There are a lot of systems implemented by other institutions that are somehow related to PABS. In recent years much effort has gone into the development of systems that can provide automatic feedback for programming assignments. Every system focuses on different aspects of improving teaching of programming skills, like user/group management, easy submission, testing/debugging, code review and feedback[Ih10].

In the following section we review a selection of tools, regarding technical and design aspects as well as core features. Some of the tools, e.g. VIPS[GW13] are completely web-based including a browser-based code editor. The advantage is that students don't need to install other tools making it easier to concentrate on solving the assignments. Other tools, e.g. MARMOSET[Sp06], also integrate directly into IDEs by providing plugins. This usually limits the set of supported IDEs. Getting in touch with development tools, especially IDEs, is a core competency for programmers, which is why we decided to use existing local applications instead of a web-based solution.

Most tools provide programming tasks for Java[MS13][ES13], but there are tools for other languages. VIPS[GW13] supports Prolog, Lisp, Haskell and Octave, the system described in [KJ13] supports SQL. MARMOSET[Sp06] supports Java, C, Ruby and Caml.

M. Novák and M. Biňas describe a system[NB11] that is completely built around existing tools. IDEs, like Netbeans or Eclipse, are used for development as well as for uploading submissions. The uploading is done utilizing ANT scripts. The second part of the system is a web service that applies different checks to the submission, like plagiarism detection. Then feedback is given through Moodle or the IDE.

Most of the tools provide testing and/or debugging abilities, e.g. ProgTEST[DMB11] or PASS[LLY10]. They use unit tests to check code functionality and give test results as feedback to the student. The student may resubmit a revised version until tests finally pass. An extension to PASS, ADA[La08] (automatic debugging assistant), tries to give

students hints for debugging their code. Web-CAT extends this functionality by not only providing tests but also having students write and submit their own tests.

In contrast MARMOSET[Sp06] tries to prevent being used as a debugging tool by using release tokens. Each student has a limited number of such tokens per day, therefore starting to work on the task early leads to more submission possibilities. MARMOSET is built as a web service utilizing Apache Tomcat and a MySQL database. Several build servers are used for security and scalability reasons. A build server downloads a submission from the submit server, checks it and sends feedback back to the submit server.

Automatic assessment is done in various tools, e.g. in Web-CAT[EPQ08], in VIPS[GW13] or in Praktomat[KJ13]. Other tools, e.g. GATE provide possibilities for manual assessment or additional manual improvement of automatic assessment (Web-CAT). Most systems deal with security issues by using Java Security Manager and having a timeout for programming code with endless loops.

3 Technical Description of the PABS System

At the University of Würzburg the number of students in computer science and related fields has drastically increased in recent years. Consequently the time needed by teachers to help students and grade assignments has also increased. PABS was written to provide students with automatically generated feedback for programming assignments and reduce the amount of time needed to tutor students. Primarily it is intended as a tool to support students attending the practical programming course but nowadays it is also used in various other courses that hand out programming assignments.

PABS is centered around the notion of a course. A course consists of an arbitrary number of assignments and has a number of students who subscribe to it. Every assignment must be solved within a configurable time that needs to be specified by a teacher. Afterwards students can still work on the assignment but will not be able to submit a final solution that will be graded. For every assignment a minimum requirement needed to submit a final solution is set by a teacher. An example requirement would be the passing of all required functional tests or the successful compilation of both solution and test code.

A web application gives students access to the assignment text and shows a list of all submissions made by a user. It also enables a user to request feedback for a submission and subsequently view the automatically generated feedback as can be seen in Figure 1. If the automatically generated feedback meets the requirements specified by the teacher, the student is then able to submit a final solution that will be graded.

PABS uses Subversion[1](SVN) to manage assignment text, configuration and tests as well as student submissions. To that end a SVN repository is created for every course. The repository seperates the configuration from the students submission using different paths. In order to prevent students from accessing the assignment configuration, tests or other

[1] `https://subversion.apache.org/`

Figure 1: Students can see which tests they passed and which they failed

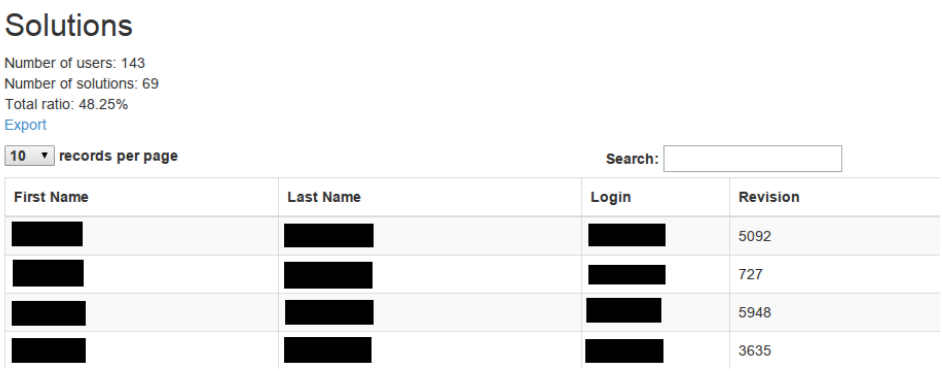

Figure 2: Administrators are shown which students already handed in successful solutions (names blurred for data protection)

students' files, the access for every student is being restricted to a single directory within the repository. Students commit possible solutions of the assignments to their path within the repository. The revisions can then be viewed and used within the web application.

The SVN repository gives students and teachers convenient access to the work history. This enables students to discard wrong solutions and go back to an earlier version without fear of losing data. Teachers on the other hand profit from being able to inspect the work history of a student to provide fast and tailored feedback as well as detect plagiarism. There are also technical advantages compared to storing submissions e.g. in a database. A SVN repository is very compact and even courses with several thousand commits only need about 100 MB of disk space. Also the repository can be setup independently from the web application which means that it can still be used in case of an outage of the web application. This setup allows the students to freely choose the used IDE or even pass on using one at all. The learning curve for SVN is softened by providing a simple reset solution in case of SVN conflicts.

Feedback generation can be a time and resource consuming task depending on the number and complexity of tests but also on code quality. To support fast feedback generation with

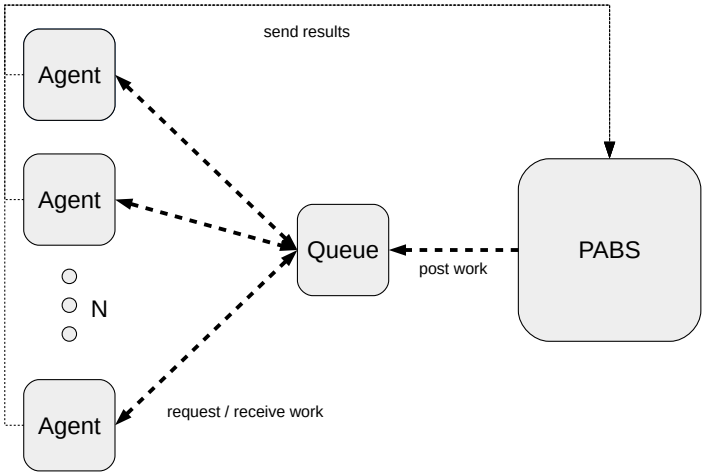

Figure 3: Shows the asynchronous communication between PABS and its agents through a queue. First PABS posts a submission into the queue. A free worker than asks the queue for work. After the worker is finished it sends the results back to PABS and informs the queue that the submission has been processed.

a growing user base PABS employs separate processes called *agents* that are able to communicate with the web application over a network. Agents can be deployed on separate machines and activated independently. While running, an agent is in constant communication with the queue of submissions maintained by the web application as shown in Figure 3. An enqueued submission is assigned to the first available *agent* available who subsequently starts processing it. After the feedback generation is completed the *agent* sends the results back to the web application and the queue is being informed that the *agent* is now available.

The communication between web application and *agent* is implemented using Akka[2]. First an *actor* responsible for managing the queue is started. Then an actor that is able to send submissions to the queue is brought up. Next an arbitrary number of *agent actors* is started and registered with the queue actor. After that the pipeline for processing is initialized and can start processing submissions. It is worth mentioning that *agents* send the generated feedback directly to the *actor* who sends the submission, thus bypassing the queue.

The *agent* utilizes Gradle[3] to compile the student's submission and execute tests and other analysis tools. Meanwhile a custom plugin gathers information about the build and makes the results available to the *agent*. The *agent* monitors the build process and terminates it if necessary e.g. because of an endless loop in the student's code. After successful execution of the build, the *agent* creates feedback data from the build information and sends it back to the web application.

[2] `http://www.akka.io`
[3] `http://www.gradle.org`

4 Practical Application

The initial vesion of PABS was developed in 2010 and introduced for the practical programming course of computer science students at the University of Würzburg, replacing an older system called "Praktomat" which back then required manual upload of the student submissions in an archive file and did not perform well under heavy load. This course is offered in every semester break and is obligatory for all students either aiming for a bachelors degree in either computer science or aerospace computer science or a teaching degree for computer science. It consist of three phases. They are accompanied by practice tutorials in which tutors answer questions of the participants.

In the first phase students are given time to learn the basics of the Java programming language if they didn't take the introductory lecture to programming.

During the second phase, the students have to complete three medium sized programming tasks, each estimated at 30 hours of work. Their solutions are automatically validated using PABS. Only GUI assignments are tested by tutors for their functionality since previous experiences with automated GUI testing have shown to be prone to errors. Also very severe restrictions on the GUI design had to be made limiting the students freedom in designing solutions. The tutors are supplied with the results of the remaining automatic tests. Only students passing all automatic tests and having their GUI functionality verified are allowed to the third phase.

In the third phase a larger more complex programming task has to be completed. To hand in the task students again have to pass the unit tests provided by PABS but this time the code of the solutions that pass the automatic tests are also checked by tutors for code style and documentation as well as for code written to cheat the unit test without writing the actual code logic. Students passing the phase's automatic and manual tests are allowed to the final task. Students that pass the third phase but not the final task are exempted from retaking the second phase when repeating the course.

The final task is primarily designed to validate that the students completed the programming tasks on their own and didn't copy from other students or even payed someone to do their tasks. Therefore the students have to complete a given task within 90 minutes with only the Java API documentation given as aid. The students can evaluate their progress by executing unit tests. The final task is graded manually.

The current major revision of PABS has been utilized for four repetitions of the practical programming course. Therefore it was possible to create some first statistics. On average 130 Students registered for the course every semester and about 51 successfully hand in solutions to all exercises in the second phase. This can be explained due to the fact, that many students that are not actually planning to take the course nevertheless register to take a peak at the exercises to prepare for future semesters. Other students may think they are capable of skipping the self study phase resulting in them not being able to hand in valid solutions. After the second phase about ten students that are repeating the course and are not required to complete phase two usually join in. Of these now 60 students admitted to the third phase about 52 students or 87 percent hand in solutions that pass the automatic

tests. Of these about 48 students or 93 percent pass the manual control steps. In the final task about 38 students or 77 percent receive passing grades and complete the course.

Those statistics prove the value of automated testing since the quota of students passing the final task is at level or above similar courses. Therefore the automatic screening in the second phase is competitive with the manual approach but requires far less manpower. The amount of tutor involvement was reduced from over 700 man hours to mere 300 man hours including the time frames for the tutorials and the preparation of exercises and unit tests. The actual amount of time necessary for the manual correction of the presented solutions has been reduced to about 80 man hours.

With PABS already having reached the second iteration of the third major version it has been stable enough that the weakest link of the automatic grading process is not the system itself which is accountable for less than 1% of all failures e.g. crashes of the sheduler. Most failures come from poorly written tests provided by the assignment creators or from students deliberately ignoring instructions. The former issue is going to be addressed by increasing the focus on testing in the education of future co-workers.

After the success of PABS the internal application has been extended to multiple other courses and lectures. Until now PABS has been used for 19 courses and has acquired almost 1300 student users. The user community at the university is steadily growing as can be seen in Table 1.

Course Name	PABS introduction	no. of repetitions	Avg. Participants
Pre-Course Programming	S13	3	161
Practical Programming Course	S13	5	130
Algorithms and Data Structures	W13	2	400
Introduction to Computer Science	W14	1	111
Practical Programming Course for Business Informatics	W13	5	159
Programming Basics	W14	1	168
Computer Science for all Departments	W14	1	86
Object Oriented Design and Java Programming	W14	1	46

Table 1: Lectures utilizing PABS system (W = Winter Term, S = Summer Term). All lectures except of the practical programming courses are offered only every second semester, the practical programming courses are offered every semester. State up to S15

5 Conclusion

We introduced PABS, a system that automatically generates feedback for programming assignments and gave an overview over related work. PABS is highly scalable and can adapt to a growing number of courses and students. In practical application PABS has shown to be highly reliable and a valuable tool for teachers offering courses with programming assignments. The amount of required man hours for student assistants has been vastly reduced. The success led to the successful introduction in multiple courses across the Department of Computer Science. The time spent on developing the PABS system has been proven valuable for the complete Department on the one hand allowing the student

assistant resources to be used for other applications and on the other hand providing and objective way to evaluate solutions handed in by the students.

Further development steps are planned like the integration of PABS in the learning managment system moodle as well as the development of grader agents to support additional languages. An open source distribution is currently not planned.

For further research there are ideas of performing a didactical evaluation of the feedback provided by PABS as well as the evaluation of different algorithms to utilize the history for better plagiarism detection.

References

[DMB11] De Souza, Draylson Micael; Maldonado, Jose Carlos; Barbosa, Ellen Francine: ProgTest: An environment for the submission and evaluation of programming assignments based on testing activities. In: Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on. IEEE, p. 1–10, 2011.

[EPQ08] Edwards, Stephen H; Perez-Quinones, Manuel A: Web-CAT: automatically grading programming assignments. In: ACM SIGCSE Bulletin. volume 40. ACM, p. 328–328, 2008.

[ES13] Eggert, Daniel; Schulze, Eike Clas: GWT basiertes System zur automatischen Bewertung von Programmieraufgaben. In: ABP. 2013.

[GW13] Gust, Helmar; Werner, Nadine: Automatische Bewertung von Übungsaufgaben in VIPS. In: ABP. 2013.

[Ih10] Ihantola, Petri; Ahoniemi, Tuukka; Karavirta, Ville; Seppälä, Otto: Review of Recent Systems for Automatic Assessment of Programming Assignments. In: Proceedings of the 10th Koli Calling International Conference on Computing Education Research. Koli Calling '10, ACM, New York, NY, USA, p. 86–93, 2010.

[KJ13] Kruse, Marcel; Jensen, Nils: Automatische Bewertung von Datenbankaufgaben unter Verwendung von LON-CAPA und Praktomat. In: ABP. 2013.

[La08] Lam, Maria SW; Chan, Eric YK; Lee, Victor CS; Yu, Yuen-Tak: Designing an automatic debugging assistant for improving the learning of computer programming. In: Hybrid Learning and Education. Springer, p. 359–370, 2008.

[LLY10] Law, Kris MY; Lee, Victor CS; Yu, Yuen-Tak: Learning motivation in e-learning facilitated computer programming courses. Computers & Education, 55(1):218–228, 2010.

[MS13] Müller, Oliver; Strickroth, Sven: GATE-Ein System zur Verbesserung der Programmierausbildung und zur Unterstützung von Tutoren. In: ABP. 2013.

[NB11] Novák, M; Biňas, M: Automated testing of case studies in programming courses. In: Emerging eLearning Technologies and Applications (ICETA), 2011 9th International Conference on. IEEE, p. 157–162, 2011.

[Sp06] Spacco, Jaime; Hovemeyer, David; Pugh, William; Emad, Fawzi; Hollingsworth, Jeffrey K; Padua-Perez, Nelson: Experiences with marmoset: designing and using an advanced submission and testing system for programming courses. ACM SIGCSE Bulletin, 38(3):13–17, 2006.