Integrating CrowdSorcerer: Lessons Learned

Nea Pirttinen
nea.pirttinen@helsinki.fi
University of Helsinki

Juho Leinonen
juho.leinonen@helsinki.fi
University of Helsinki

ABSTRACT

CrowdSorcerer is a tool for crowdsourcing programming assignments and teaching testing. While originally developed for the introductory Java courses at the University of Helsinki, the tool is currently being integrated into the introductory Python course at the University of Toronto. Our goal is to make CrowdSorcerer easily integrable for multiple types of materials and contexts. This facilitates the crowdsourcing aspect by both increasing the size of the population using the tool as well as diversifying the content that is generated.

CCS CONCEPTS

• **Information systems** → *Crowdsourcing*; • **Social and professional topics** → *Computing education.*

KEYWORDS

crowdsourcing, assignment creation, testing

ACM Reference Format:
Nea Pirttinen and Juho Leinonen. 2019. Integrating CrowdSorcerer: Lessons Learned. In *Proceedings of SPLICE '19*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Crowdsourcing is well-suited for big tasks that can be split into multiple smaller tasks. Instead of focusing a huge workload for one person, a group of people can create the same outcome by doing only a little bit of work each. Crowdsourcing has been successfully used for example in Amazon Mechanical Turk[1], a website through which businesses can hire users to complete small tasks that require manual work. Another example of wide-scale crowdsourcing effort is Wikipedia[2], a multilingual online encyclopedia maintained through users' open collaboration.

Crowdsourcing has been proven to be a working collaboration method also in education. For example, online material collection OpenDSA [6] has been developed by crowdsourcing and supports a wide range of computer science courses, such as data structures and algorithms, programming languages, and formal languages with tools such as visualizations and interactive exercises. PeerWise [1] is a web-based tool used by students of various fields, including computer science, used to create and thus crowdsource multiple choice questions. Relying on computer science educators rather than student efforts, Canterbury QuestionBank [5] is a comprehensive set of multiple choice questions suitable for first-year computer science courses.

One key requirement of crowdsourcing is having a big crowd. In the educational context, a single university can usually only offer a relatively small crowd, especially since crowdsourcing efforts are generally focused on a specific task on a single course, or a handful of courses at best. Multiple collaborating universities can offer a bigger crowd, as well as multiple kinds of contexts to investigate.

This raises a question: How should a crowdsourcing tool be developed so that it is interoperable? We have originally built CrowdSorcerer for the University of Helsinki introductory Java course, held in Finnish, and are now integrating the tool to the University of Toronto introductory Python course, held in English.

2 OVERVIEW OF CROWDSORCERER

CrowdSorcerer [3] is a programming education tool designed to be easily embeddable into any online course material. The students using the tool are expected to create a programming assignment of their own from scratch, using the instructions given by the course instructor as a basis (such as "Create an assignment that uses for-loops"). The students come up with an assignment handout describing the problem that is to be solved, a source code that is divided to a code template and a model solution, and test cases for the program. The code template acts as a structure and can include for example the method declaration or a main method. The model solution only includes the functionality of the program, that is, the lines that a student trying to complete the assignment would need to write into the code template. The students also need to provide tags that describe their assignment, such as "for-loop", "if-else" or "very difficult". These are to help with the filtering of the exercise database if the student-created assignments are to be used in course materials.

When submitted, the assignment is tested with the student-provided test cases, and possible compilation errors or test related errors are given to the student to investigate if need be. Students can create test cases in three different ways, each requiring a different level of understanding of testing practices.

(1) **Input-output tests:** The student gives the input for the program, and the expected output for that specific input.
(2) **Input-output with visible unit test source code:** The student gives the name of the test case, which assertion type to use (*contains*, *does not contain* or *equals*), and input-output pairs similarly to the first case. These are inserted into the source code of the unit test as the student changes the values so that they can see how their changes affect the source code of the unit test.
(3) **Write a complete test method:** The student writes the whole unit test.

[1] https://www.mturk.com/
[2] https://www.wikipedia.org/

SPLICE '19, August 11, 2019, Toronto, CA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Completed assignments with fully passed tests can be peer reviewed within the same tool [4]. When reviewing, the students see the assignment in full, and they can investigate the code template and the model solution separately. Reviewing is done through statements such as "The code template and the model solution are separated correctly" and "The test coverage is reasonable", which are rated on a Likert-like scale ranging from one to five. The students are also expected to give a short written review, as well as tags that they think suit the assignment the best.

Though this setup is what we have used in our introductory Java courses in the University of Helsinki, all the components of CrowdSorcerer are modifiable. For example, we are currently integrating the tool to an introductory Python course at the University of Toronto, where the focus is on teaching testing instead of creating assignments. In this case, the students do not write an assignment handout, give tags or edit the source code. Instead, the tool presents a ready-made program (created by the instructor) that cannot be edited. The students are only expected to write test cases for the existing program. Currently, CrowdSorcerer supports testing types (1) and (2) for this kind of practice, and type (3) is under development.

3 LESSONS LEARNED

CrowdSorcerer was originally developed for the Java programming course materials at the University of Helsinki. The tool has been used on multiple programming courses since autumn 2017. The material base has gone through some major updates during the operation of CrowdSorcerer, but we have been able to keep the tool online throughout the material update processes. This is a benefit that comes from a microservice architecture – since CrowdSorcerer functions as its own unit outside the material it is embedded to, only the components that handle the importing need to be updated.

Currently, CrowdSorcerer can be imported into an online course material in two different ways: either as a React component that can be installed through npm or through an HTML script tag. The University of Helsinki course material currently uses the first approach, while the University of Toronto course material has taken the latter.

At the time of writing this paper, CrowdSorcerer is being integrated into the University of Toronto Python material. We will report more on the lessons learned as the integration finishes.

4 GOALS FOR THE FUTURE

Future goals for CrowdSorcerer are twofold: collecting a large pool of programming assignments, and helping students learn testing in a straightforward way.

The programming assignment database offers the possibility of enhancing adaptive content in programming course materials. Reviewed assignment could be used, for example, to give students who struggle with a particular topic of the course some extra practice to help them stay on track. An open pool of assignments in multiple programming languages and easy integration of those assignments to different types of programming materials would ease the load that instructors have while creating and updating their course materials.

While research so far on whether CrowdSorcerer helps students to learn testing has given debatable results [2], we hope that improvements made to the tool give the students a better learning experience. After integrating the tool to the materials at the University of Toronto, our goal is to study whether using CrowdSorcerer helps students understand and learn testing more efficiently. A study will be conducted during autumn 2019, and our intent is to conduct the study both at the University of Helsinki and the University of Toronto to measure the effects in various contexts.

ACKNOWLEDGMENTS

We are grateful for the support provided by the SPLICE project for integrating CrowdSorcerer at University of Toronto.

REFERENCES

[1] Paul Denny, Andrew Luxton-Reilly, and John Hamer. 2008. The PeerWise System of Student Contributed Assessment Questions. In *Proceedings of the Tenth Conference on Australasian Computing Education - Volume 78 (ACE '08)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 69–74. http://dl.acm.org/citation.cfm?id=1379249.1379255

[2] Vilma Kangas, Nea Pirttinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2019. Does Creating Programming Assignments with Tests Lead to Improved Performance in Writing Unit Tests?. In *Proceedings of the ACM Conference on Global Computing Education (CompEd '19)*. ACM, New York, NY, USA, 106–112. http://doi.acm.org/10.1145/3300115.3309516

[3] Nea Pirttinen, Vilma Kangas, Irene Nikkarinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Crowdsourcing Programming Assignments with CrowdSorcerer. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*. ACM, New York, NY, USA, 326–331. http://doi.acm.org/10.1145/3197091.3197117

[4] Nea Pirttinen, Vilma Kangas, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Analysis of Students' Peer Reviews to Crowdsourced Programming Assignments. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling '18)*. ACM, New York, NY, USA, Article 21, 5 pages. http://doi.acm.org/10.1145/3279720.3279741

[5] Kate Sanders, Marzieh Ahmadzadeh, Tony Clear, Stephen H. Edwards, Mikey Goldweber, Chris Johnson, Raymond Lister, Robert McCartney, Elizabeth Patitsas, and Jaime Spacco. 2013. The Canterbury QuestionBank: Building a Repository of Multiple-choice CS1 and CS2 Questions. In *Proceedings of the ITiCSE Working Group Reports Conference on Innovation and Technology in Computer Science Education-working Group Reports (ITiCSE -WGR '13)*. ACM, New York, NY, USA, 33–52. http://doi.acm.org/10.1145/2543882.2543885

[6] Clifford A. Shaffer, Ville Karavirta, Ari Korhonen, and Thomas L. Naps. 2011. OpenDSA: Beginning a Community active-eBook Project. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research (Koli Calling '11)*. ACM, New York, NY, USA, 112–117. http://doi.acm.org/10.1145/2094131.2094154