International **R**eview of **R**esearch in **O**pen and **D**istributed **L**earning
Volume 19, Number 4

September – 2018

Troubleshooters for Tasks of Introductory Programming MOOCs

Marina Lepp, Tauno Palts, Piret Luik, Kaspar Papli, Reelika Suviste, Merilin Säde, Kaspar Hollo, Vello Vaherpuu, and Eno Tõnisson
University of Tartu, Estonia

Abstract

Learning programming has become more and more popular and organizing introductory massive open online courses (MOOCs) on programming can be one way to bring this education to the masses. While programming MOOCs usually use automated assessment to give feedback on the submitted code, the lack of understanding of certain aspects of the tasks and feedback given by the automated assessment system can be one persistent problem for many participants. This paper introduces troubleshooters, which are help systems, structured like decision trees, for giving hints and examples of certain aspects of the course tasks. The goal of this paper is to give an overview of usability (benefits and dangers) of, and the participants' feedback on, using troubleshooters. Troubleshooters have been used from the year 2016 in two different programming MOOCs for adults in Estonia. These MOOCs are characterized by high completion rates (50–70%), which is unusual for MOOCs. Data is gathered from the learning analytics integrated into the troubleshooters' environment, letters from the participants, questionnaires, and tasks conducted through the courses. As it was not compulsory to use troubleshooters, the results indicate that only 19.8% of the users did not use troubleshooters at all and 10% of the participants did not find troubleshooters helpful at all. The main difference that appeared is that the number of questions asked from the organizers about the programming tasks during the courses via helpdesk declined about 29%.

Keywords: MOOC, open education, programming, troubleshooting system

Introduction

Teaching introductory programming courses has become an important subject matter in Estonia in connection with the need to raise awareness of, and interest in, information technology. Supporting the learning of the programming language Python, a massive open online course (MOOC) in Estonia called *About Programming* (in Estonian, *Programmeerimisest maalähedaselt*) was created in 2014. Research has shown that the average completion rate for MOOCs in the world is approximately 15% (Jordan, 2014; Siemens, 2013), but in our case the percentage of completions has been constantly over 50%. This paper addresses the idea of having a helpdesk supporting the participants in the course and reducing the number of questions from the participants by creating troubleshooters for the programming tasks.

Programming MOOCs rely mostly on automated assessments, which enable the participants to post the solutions for the tasks in a way that the system could automatically analyze the solutions and give automated feedback. Self-assessment should be used as an assessment for learning instead of an assessment of learning (Admiraal, Huisman, & Pilli, 2015). In programming, some mistakes in the code can be very difficult to resolve and therefore our MOOCs offered a helpdesk email address to answer the questions that appear during the course. The instructors and university students who lent their assistance, agreed to answer the helpdesk emails in less than 8 hours. While having people on watch all the time is not very cost effective, the helpdesk offers instant help that beginner learners need. The questions asked from the helpdesk give a lot of information about the problems occurring with the tasks during the course.

To reduce the number of questions asked from the helpdesk, troubleshooters were provided for every programming task, starting from 2016. The troubleshooters include collections of answers and clues to the questions, which can arise when solving the course tasks.

This paper gives an overview of the creation of the troubleshooters to support the course and presents the learners' opinions about the troubleshooters. The impact of troubleshooters is discussed in the context of the resources needed for creating troubleshooters and the results of course optimization, needed to keep it automated.

Theoretical Background

This section provides a theoretical background on supporting online programming courses with helpdesk and troubleshooters by categorizing programming mistakes that beginners make.

MOOCs

Massive Open Online Courses (MOOCs) are one of the recent models in open and distributed learning (Downes, 2017). The history of MOOCs can be divided into two phases: cMOOC (connectivist MOOCs) period and xMOOC (content-based MOOCs) period (Baturay, 2015). However, there is a move away from the cMOOC/xMOOC division towards recognition of the multiplicity of MOOC designs, purposes, topics, and teaching styles (Admiraal et al., 2015).

While the educational world is proliferated with MOOCs and they are hyped in the media, there are still some challenges for MOOCs to overcome (Veletsianos & Shepherdson, 2016). One of the most salient challenges is the dropout rate (Siemens, 2013), with widely cited figures of 10% completion rates (Ebben & Murphy, 2014). Researchers are trying to examine the reasons behind the low retention rates (Greene, Oswald, & Pomerantz, 2015; Hone & El Said, 2016). It has been found that a lack of incentive, insufficient prior knowledge about the topic, ambiguous assignments, and having no one to turn to for help can be possible reasons for non-completion (Hew & Cheung, 2014). MOOC content and interaction with the instructor were also shown to have a significant effect on retention (Hone & El Said, 2016).

Due to having thousands of participants per instructor, it is impossible for MOOC instructors to conduct assessments and provide individual feedback (Suen, 2014). Different models of interaction are used, such as automated feedback (Pieterse, 2013), peer support (Onah, Sinclair, & Boyatt, 2014), self-assessment (Papathoma, Blake, Clow, & Scanlon, 2015), helpdesk (Warren, Rixner, Greiner, & Wong, 2014), and scaffolding messages like troubleshooters (Vihavainen, Luukkainen, & Kurhila, 2012).

Helpdesks

As the number of questions on various topics of the course rises and it is difficult to find answers to the questions in a course with thousands of participants, we were faced with the challenge of how to retain the availability of sufficient support to positively finish the course. Using a helpdesk could be one option for answering the questions and monitoring the process. Previous MOOCs that used a helpdesk were rated extremely positive (Warren et al., 2014).

A helpdesk could use different kinds of data, video, and voice support (Motsuk, 1999), but our course offered a helpdesk email from the organizers of the MOOCs (faculty members and students) who had to answer any letters in less than 8 hours. The possibility to ask questions from the helpdesk could have been one of the key factors that helped more than 50% of the participants finish our courses (Lepp et al., 2017a).

As course participants send emails to the helpdesk address and receive answers from it, several helpdesk systems are available for managing such a system. A helpdesk system needs to be usable online, look nice and simple for users, be easy to use, include various functions, like a search engine, option to set labels to letters, and archive the letter data for later analysis. Developing such a system can be too complex task for a simple project (Washburn & El-Bayoumi, 2003). In our case an online helpdesk system, called Freshdesk (https://freshdesk.com/) was used.

Using a helpdesk has several advantages for organizers, too. One of the benefits is that engaging students in answering the helpdesk emails can have a positive influence on their studies (McRitchie, 2009) and reduce the cost of helpdesk (Sinnett & Barr, 2004). When counting the number of people getting help and being educated by MOOCs, the cost per participant can be rather low too. Frequently asked questions can be gathered to create helpful troubleshooters for each course task.

Troubleshooters

Troubleshooters are systems that are mostly used for IT services helping to solve problems manually by clicking answers to various questions to find a solution to the problem in a system with a decision tree structure. A similar kind of self-assessment (exercises with built-in scaffolding messages inside the programming environment) has been tried in case of programming MOOC and found to be fruitful (Vihavainen et al., 2012).

One way of identifying the problems that need to be included in troubleshooters would be mining the course data (and constructing, for example, Bayesian networks; Skaanning, Jensen, & Kjærulff, 2000). It can be difficult, as many filters should be applied to get reasonable results (Wustner, Joumblatt, Teixeira, & Chandrashekar, 2012). Sometimes the problems occurring can be rather difficult to track, as the real problems can be different from those originally discovered.

Creating troubleshooters can be difficult, but systematically organizing the problems that need to be solved can make it a lot easier. The presence of the course personnel in labs can be one possibility for answering the question about the next problem that can be encountered by a student (Vihavainen et al., 2012). In case of MOOCs, creating systematic decision trees for troubleshooters can be done by analyzing past help requests for the tasks and categorizing the questions in a way that supports the development of hints and examples to guide learners to answers to frequently asked questions.

Categorizing the Problems in Solving Programming Tasks

This paper addresses the system of help for typical problems of novice programmers. As many questions arise during the programming MOOCs, starting from questions about registration and ending with understanding specific nuances of certain aspects, this article is limited to the frequently asked questions that have been asked by the participants in an introductory programming MOOC. It can be much more difficult to help with the problems in more complex courses, including aspects such as inheritance, objects, class, encapsulation, methods, message passing, polymorphism, and abstraction (Sanders & Thomas, 2007).

Many questions can be about error messages. The Center for Computing Education and Diversity at the University of California has identified 17 categories of errors that can occur in Python programming (Marceau, Fisler, & Krishnamurthi, 2011), but when looking at one task, few of them usually occur and users are often accustomed to that when trying to resolve a mistake in the code. Error messages are only a part of the problems that can occur and code can often be wrong even when executed with no errors. This could be the case, for example, when trying to understand the changes that need to be made in the code to produce different outputs for certain inputs.

Garner, Haden, and Robins (2005) have organized introductory programming courses and investigated the mistakes novice programmers make during the practice sessions. They noticed that the more assistance weaker participants receive the better is their achievement in the course. Garner and colleagues described 27 problems that can appear in the practice sessions of a programming course for beginners. As our courses were online courses, we had to use helpdesk letters instead of direct feedback from practice sessions.

The problems occurring can be different in various situations. In pair-programming, the pairs would later be able to solve more low-level syntax problems individually than in solo-programming (Hanks, 2008). As in our courses the assignments are individual, we needed a system to help more with the low-level syntax problems.

As the problems appear during the process of solving certain tasks, our idea was to cultivate from that and to look at the problems coming out from the MOOC tasks via the helpdesk. Although in our case many of the problems (like errors and input-output faults) are solvable with the help of the automatic assessment tool, that assessment tool can create extra problems and questions that need to be solved.

Research Problem

The purpose of this study was to develop and evaluate troubleshooters for the programming tasks to provide additional support to MOOC participants and reduce the number of learner emails with questions to organizers while maintaining a high completion rate. Figure 1 presents the research problem.

The research questions were:

1. Can troubleshooters facilitate the work of MOOC organizers?
2. How do participants perceive troubleshooters as an additional support tool?

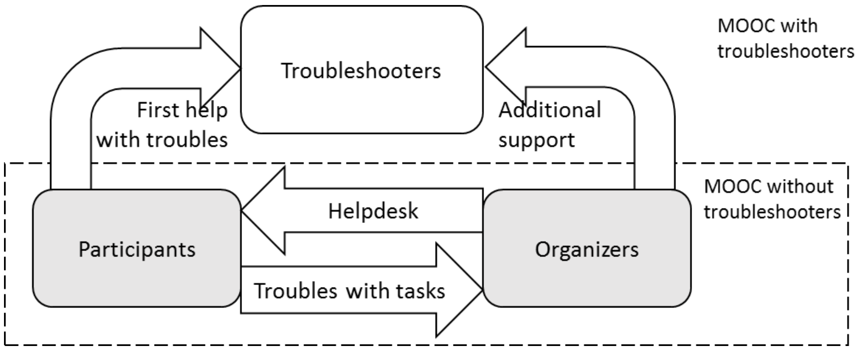


Figure 1. The research problem.

Murelahendaja Environment for Troubleshooters

Based on previous studies (Garner et al., 2005; Vihavainen et al., 2012), our troubleshooter creation process, which was rather difficult and time consuming, includes:

1. Analyzing the questions asked via the helpdesk about the weekly programming tasks;
2. Categorizing the questions asked by creating a table of types of typical questions;
3. Creating a tree-structured hint system with examples called troubleshooters to help with questions that have been asked.

Analysis of Questions and Categorization of Occurring Problems

This paper deals with an introductory programming MOOC *About Programming* in Estonia for adults that has been organized several times since December 2014. The Institute of Computer Science also organizes a MOOC named *Introduction to Programming*, which will only be touched upon briefly in this article.

A helpdesk was organized in our MOOCs to help participants with their problems and to get an overview of the questions asked about the tasks. After collecting the questions that were asked from the helpdesk in 2015, a table of data was compiled to categorize the problems that occurred in certain aspects of the tasks. This paper focuses on troubleshooters created for the course in 2016 to help with these problems with the programming tasks.

As our idea was to create helpful hints for the tasks of each week, it meant that each task needed to be looked at separately. The course *About Programming* had eight different parts in 4 weeks (2 parts per week): introduction (algorithm and program, part I), variables and data types (II), conditionals (III), strings (IV), loops (V), regular expressions (VI), functions (VII), and conclusion (part VIII). Tasks were provided only for parts II to VII. The organizers received a total of 1,250 letters with questions from 1,534 participants in the MOOC of 2015. Some letters were related to organizational issues. The statistics for parts II to VII show that most of the questions were asked about the task of part VII (see Figure 2).

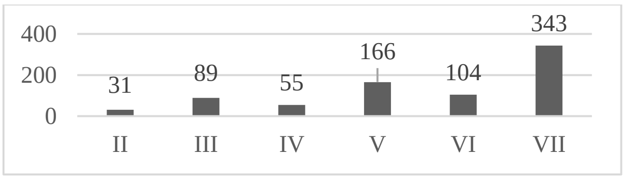


Figure 2. Number of questions asked per task from the helpdesk.

A description and a manual were created to help allocate the problems asked from the helpdesk into the categories. Letters were broken down into separate questions, each representing one problem. Questions from one letter could belong to a number of different categories. Three experts were used to evaluate 10% of the total number of problems asked from the helpdesk randomly to see if the descriptions of the categories were understood similarly. The overlap in the categorization of the problems was 80%. Most of the differences were caused by the fact that some of the questions asked from the helpdesk can lead to several problems and the letters from the participants were not that clear.

The questions were categorized based on existing classifications (Garner et al., 2005) and judgements of the course organizers. In total, 30 categories of occurring problems were discovered for the MOOC of 2015. Ten categories were related to organizational problems with registration to the course and the software used during the course. Twenty categories were related to programming tasks with the following keywords: input, datatype, variable, syntax and whitespaces, output, round, loops and conditionals, choosing symbols, using default functions, wrong order of input, calculations, iteration, finding the sum, sum vs counter, wrong regular expression, missing regular expression, module import, argument of the function, calling a function, and creating a file.

Initially, the questions were analyzed and categorized by weekly tasks. As different tasks can have similar problems, some of the categories were included in several tasks. Five to nine categories were identified per task. The categories were rather specific to the tasks to give the best help for the questions asked. The categories provided the basis for creating a troubleshooter for the particular task. Our weekly tasks and topics are mostly typical for introductory programming courses, which means that the occurring problems are also rather typical, but can also depend on the text of the particular weekly task (for example, finding the sum).

An example of the problems occurring in weekly task VII can be seen in Table 1. For this task, the aspects listed in the table need extra help from the organizers so that the troubleshooter could give hints and examples to help with those problems. As the topic of the seventh part is functions, mostly questions about using functions were asked (calling a function and argument of the function), but other categories are closely related to that topic and the task too. Several registered problems were also related to the contents of previous parts of the course (for example, variable).

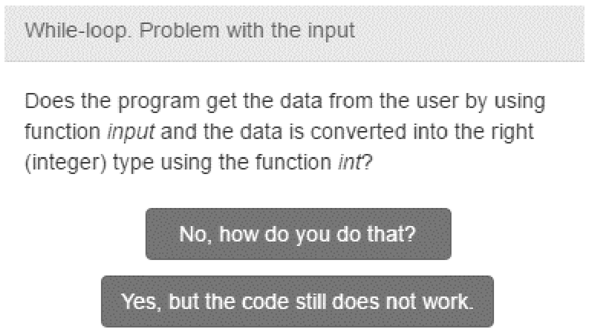
Table 1

Categories of Occurring Problems for Task VII

Keywords of the category	**Number of times occurred**
Variable	74
Calling a function	64
Round	51
Argument of the function	38
Datatype	31
Input	31
Syntax and whitespaces	23
Calculations	19

Troubleshooters

After the categorization of the problems with programming tasks was complete, the environment called *Murelahendaja* was created to offer decision-tree-structured hints and examples called troubleshooters (see Figure 3). Along the way of creating the troubleshooters, the environment was further developed. The functionalities of looking at the tree of troubleshooter and getting a statistical overview of the usage of troubleshooters were added during the development process.

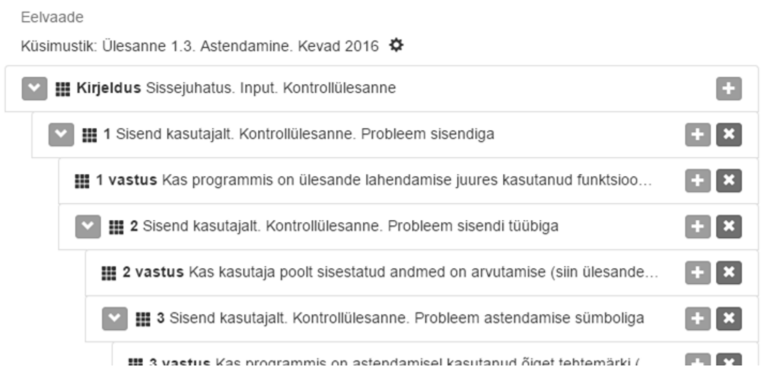

While-loop. Problem with the input

Does the program get the data from the user by using function *input* and the data is converted into the right (integer) type using the function *int*?

Figure 3. An example of a troubleshooter question.

The *Murelahendaja* environment has two separate views: i) one for registered users to create troubleshooters and view the statistics of usage, and ii) second for course participants (guests) to use troubleshooters.

Registered users can create troubleshooters by adding linked pages with questions about the problems of the tasks and helpful hints with code examples to help solve the problem. Pages are linked together in a decision tree structure and an overview of the linked pages can be seen on one screen (see Figure 4).


Eelvaade

Küsimustik: Ülesanne 1.3. Astendamine. Kevad 2016

Kirjeldus Sissejuhatus. Input. Kontrollülesanne

1 Sisend kasutajalt. Kontrollülesanne. Probleem sisendiga

1 vastus Kas programmis on ülesande lahendamise juures kasutanud funktsioo...

2 Sisend kasutajalt. Kontrollülesanne. Probleem sisendi tüübiga

2 vastus Kas kasutaja poolt sisestatud andmed on arvutamise (siin ülesande...

3 Sisend kasutajalt. Kontrollülesanne. Probleem astendamise sümboliga

3 vastus Kas programmis on astendamisel kasutanud õiget tehtemärki (

Figure 4. Overview of created troubleshooter pages.

Registered users can see descriptive statistics about the use of each page of the troubleshooters (see Figure 5). Statistics show on a tree graph how many times a troubleshooter has been viewed (letter "v") and how many times people have indicated that the hints and code examples were helpful by clicking "It worked!" (letter "s"). Figure 5 shows that the first step of the troubleshooter always includes an introduction to the troubleshooter. The second step asks from the user if the respective function is used by the user in the solution. There are two branches after that question – button "No, how do you do that?" leads to the page explaining the usage of the function in the program and button "Yes" leads to asking the next question about the next trouble.

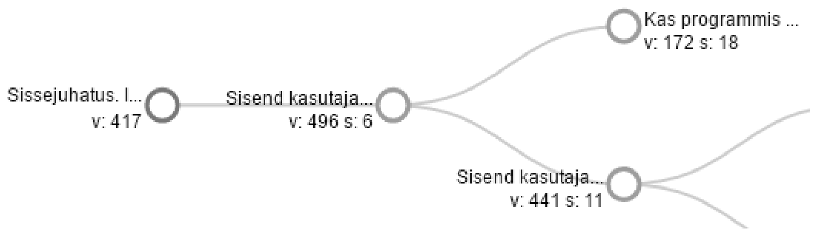
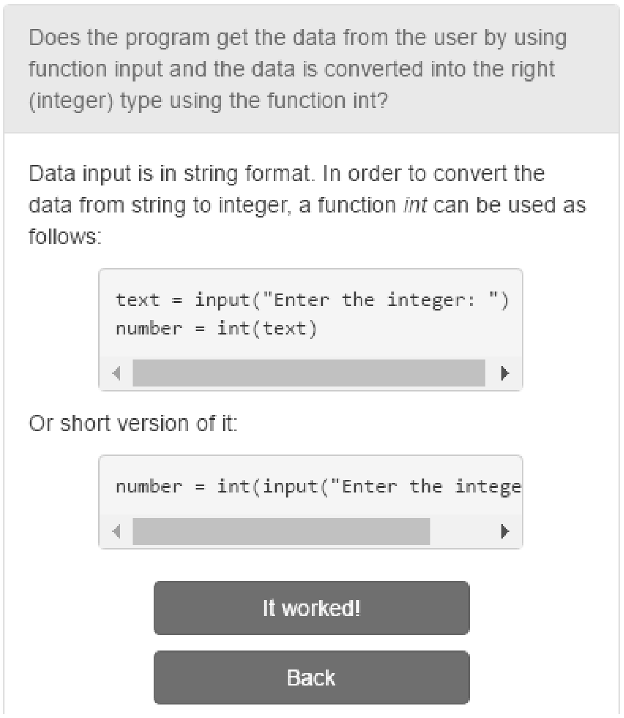


Figure 5. Tree graph with the statistics of a troubleshooter.

Guests, who are in our case the participants of the course, can see the troubleshooter as a series of questions asked one-by-one to lead to the problematic part of the task. Each question has a button to display hints and example code to solve the problem, which means that most of the questions have two buttons with the following texts to choose from (see Figure 3):

1. Button with the text "No, how do you do that?" – leads to the page with hints and examples to find an answer to the question (see Figure 6). That page has one button to go back to the question page and another button with the text "It worked!" which indicates that the clues and examples helped to solve the problem.

2. Button "Yes, but the code still does not work." – leads to the next question.


Does the program get the data from the user by using function input and the data is converted into the right (integer) type using the function int?

Data input is in string format. In order to convert the data from string to integer, a function *int* can be used as follows:

```
text = input("Enter the integer: ")
number = int(text)
```

Or short version of it:

```
number = int(input("Enter the intege
```

Figure 6. Hints and example codes for solving the problem.

Troubleshooters created in the *Murelahendaja* environment can be used in various situations in different courses (not necessarily in programming courses). For example, programming MOOCs use troubleshooters also for problems with registering to the course (see Figure 7).

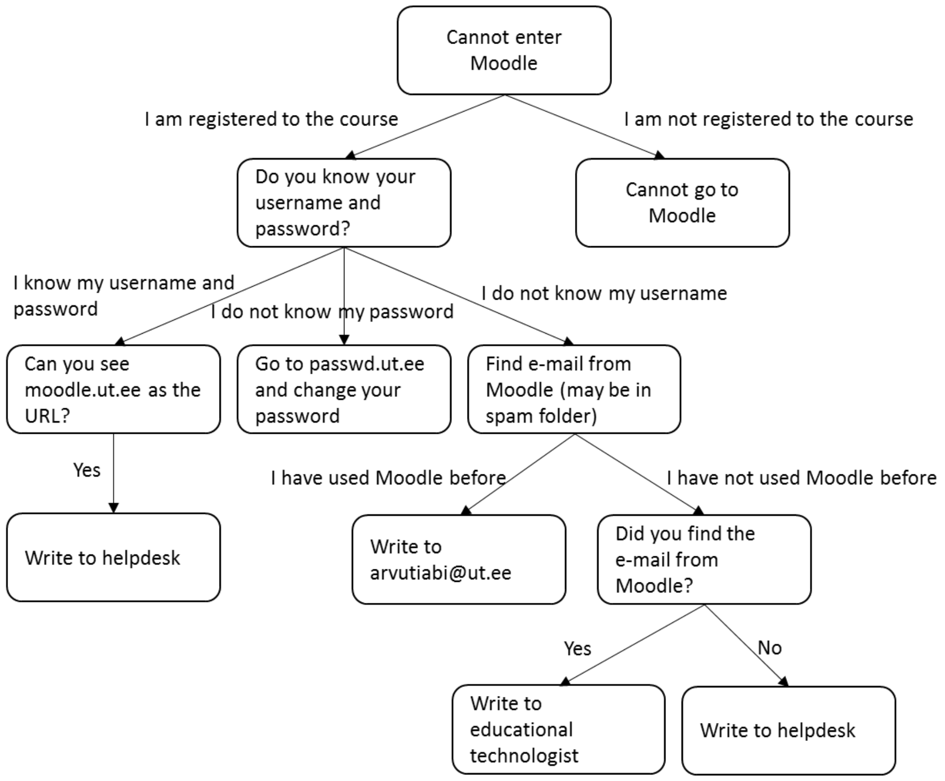

Cannot enter Moodle
I am registered to the course
I am not registered to the course
Do you know your username and password?
Cannot go to Moodle
I know my username and password
I do not know my password
I do not know my username
Can you see moodle.ut.ee as the URL?
Go to passwd.ut.ee and change your password
Find e-mail from Moodle (may be in spam folder)
Yes
I have used Moodle before
I have not used Moodle before
Write to helpdesk
Write to arvutiabi@ut.ee
Did you find the e-mail from Moodle?
Yes
No
Write to educational technologist
Write to helpdesk

Figure 7. Example of troubleshooter.

Every programming task was supported with a troubleshooter and a troubleshooter was created for each weekly task of the course. During the process of creation and usage of troubleshooters, the environment was tested and supplemented so that it would contain all the questions asked from the helpdesk in a sensible way. Troubleshooters never give a direct answer to the questions, but help with hints and examples.

As it can be difficult to navigate in a large system of hints, the troubleshooters for tasks were kept as linear as possible (Figure 8). Troubleshooters for the course *About Programming* contain 5-9 questions with 5-9 examples. Creating troubleshooters requires rather specific knowledge and experience, to identify the type of task and questions that could be helpful, and for this reason most of the troubleshooters were created by one or two persons. That guarantees that the style of troubleshooters is uniform throughout the course. The creation process takes a lot of time and energy, which means that the troubleshooters were not drastically changed for the next courses.

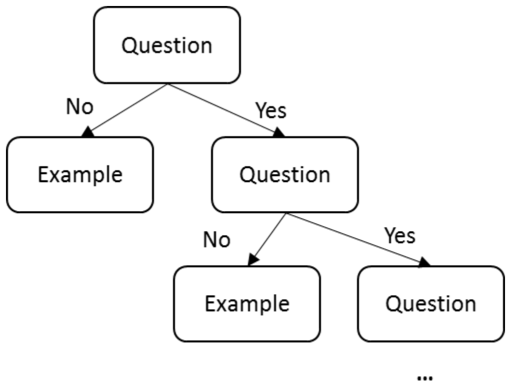


Figure 8. Structure of troubleshooter for programming task.

The technologies used for creating the web application *Murelahendaja* included CSS, HTML, JavaScript, AngularJS, D3.js, and MongoDB. The requirements for *Murelahendaja* included a web application that works in all popular web browsers and has an interface in Estonian language. The system had to be able to handle at least 1,000 guests at a time and have a response time of 0.5 seconds with the maximum response time of 2 seconds. It had to be available at least 99% of the time; critical errors had to be fixed in an hour. The *Murelahendaja* environment and user registration form can be found at progtugi.cs.ut.ee.

Potential Advantages of Troubleshooters for Online Courses

The troubleshooters may have the potential to be an additional supportive self-assessment tool in MOOCs. First, participants can use troubleshooters when they are stuck before writing to helpdesk (see Figure 9), thereby reducing the number of letters to organizers. Second, troubleshooters as part of MOOC content can have a positive effect on MOOC completion rate. In addition, troubleshooters with hints and examples can provide additional learning material and stimulate further thinking as participants study them. Furthermore, troubleshooters can be used not only in MOOCs but in traditional courses as well.

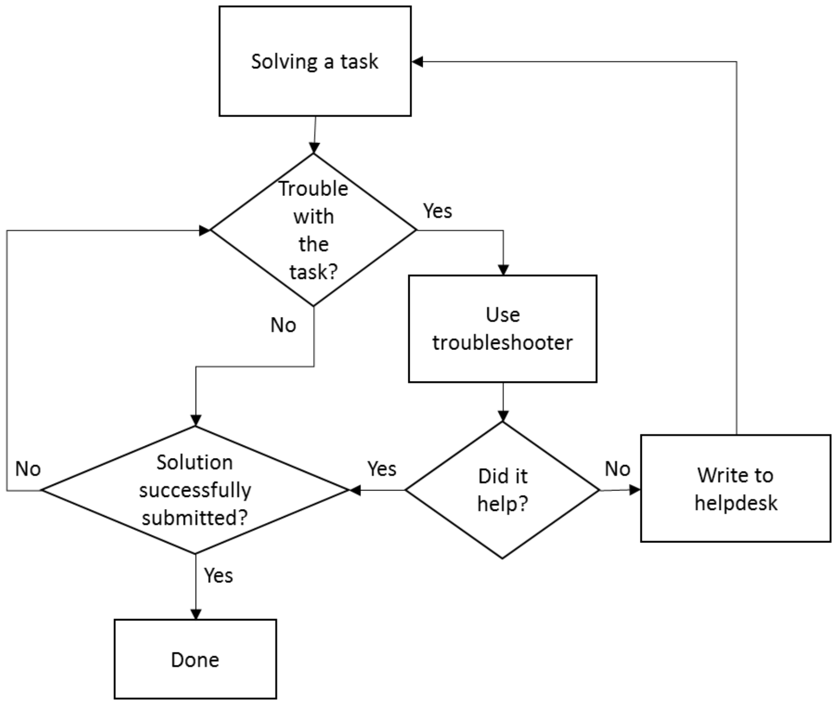


Figure 9. The role of troubleshooter in solution process.

Evaluation of Troubleshooters

The *Murelahendaja* environment for troubleshooters was evaluated to examine the effectiveness of troubleshooters in a programming course.

Research Methods

Participants and context. In autumn 2015, programming MOOC *About Programming* was held for the third time with 1,534 participants, and 1,010 (66%) of them successfully finished the course. In spring 2016, programming MOOC *About Programming* was held for the fourth time with 1,430 participants, and 885 (62%) of them successfully finished the course. The course in 2015 used a helpdesk, but no troubleshooters, which were added in 2016 (Lepp et al., 2017a). We are improving our courses gradually with new technical tools. For example, Muuli et al. (2017) describe a novel form of automated feedback. The troubleshooters were created on the basis of this MOOC and the number of questions to helpdesk was used for answering the first research question.

We collected feedback data about troubleshooters from 792 participants (89.5% of completing learners), who completed the course in the spring of 2016. From the participants 342 (43.2%) were male and 450

(56.8%) female, and 790 (99.7%) were from Estonia. The average age of the participants was 35.8 years (SD=10.9) ranging from 12 to 81.

Instruments and procedure. Data is gathered from the learning analytics integrated into the troubleshooters' environment, letters from the participants, questionnaires, and Moodle's learning analytics.

In the beginning of the courses, questionnaires were sent to get some background information about the participants and their attitude toward certain aspects, including mathematics and programming. At the end of the course, another survey was conducted to ask opinions about the course, for example, the usage and helpfulness of troubleshooters, the evaluation on the difficulty of last exercises, and the last weekly quiz. Both questionnaires were online questionnaires. The answering on these questionnaires was voluntary and passing the MOOC did not depend on that.

The Moodle learning analytics of each participant, indicating the attempts to submit tasks and the points for tasks, was matched to the answers from the questionnaire and to the background data from the pre-questionnaire.

Data analysis. Statistical analyses were carried out as follows. First, the learning analytics integrated into the troubleshooters' environment was studied. Next, descriptive statistics on the participants' opinion on using troubleshooters was investigated. Then Spearman correlation coefficients were calculated to investigate the relationship between participants' evaluations on various statements and their evaluations on the usage of troubleshooters. The helpfulness of troubleshooters for learners was also investigated using the Spearman correlation coefficient. The analyses were carried out using the statistical package SPSS version 23.0.

Results

When looking at the statistics, the total number of people clicking the button "It worked!" was 2,180 (see Figure 10). This chart shows that troubleshooters provided the most help for weekly tasks III and V. Data from the helpdesk questions from the previous course showed that the same weekly tasks prompted many questions, too. The biggest difference is that weekly task VII did not get that much help from troubleshooters as expected, but the reason could be that people had received help for many aspects from the previous troubleshooters or just did not click "It worked!" as the course was ending.

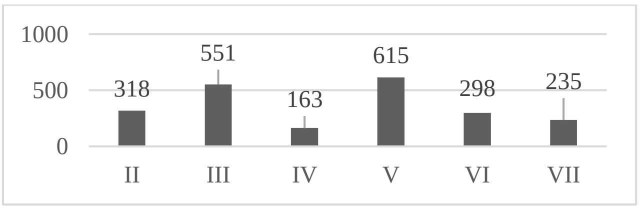


Figure 10. Number of people getting an answer from troubleshooter per weekly task.

As troubleshooters were used in the fourth instance of the course *About Programming*, the number of letters received by the helpdesk can be compared with the previous time the course was conducted. Previous time (without troubleshooters), the helpdesk received 1,250 letters from 1,534 participants, but after adding troubleshooters to the course, 750 letters were received from the 1,430 participants. There were no other major changes in the course, which means that the percentage of questions per participator declined 29%.

The MOOC *About Programming* concluded with a feedback form, which included questions about troubleshooters. The total number of people answered the final questionnaire was 792 and 635 of them had used troubleshooters. As troubleshooters were not compulsory, the results indicate that 16.6% of the participants did not look at troubleshooters at all (see Figure 11).

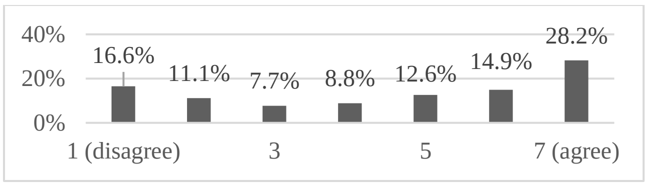


Figure 11. Looked at troubleshooters.

When the users (n=635) were asked about the helpfulness of troubleshooters, 40.8% of the participants claimed troubleshooters to be very helpful (see Figure 12) and 3.5% of the participants did not find troubleshooters helpful at all.

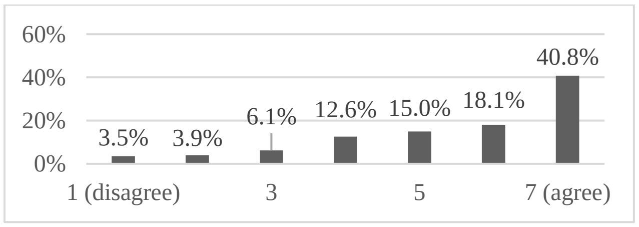


Figure 12. Got help from troubleshooters.

One of the questions included ordering the various parts of the course (videos, reading materials, extra materials, stories, tasks, forum, test, troubleshooters, and other materials) by their position in the solving process when they were used. According to the final questionnaire, 19.8% of the participants did not use troubleshooters at all and 7% looked at troubleshooter as the last thing in the process of solving the tasks (see Figure 13). For the rest of the users, troubleshooters were located at various places in the order of resources. For example, some participants used troubleshooters even before solving the weekly tasks, which means that troubleshooters have changed the way course participants learn.

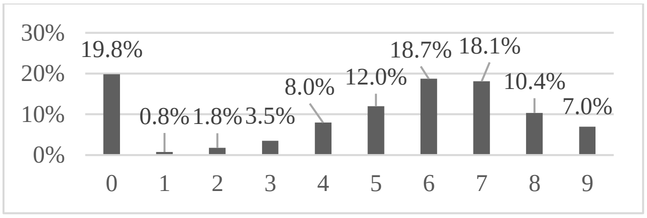


Figure 13. Place of troubleshooters in the order of solving.

Using troubleshooters correlated with various aspects of the course (see Table 2; Lepp et al., 2017b). In the beginning of the course, the participants had to answer on the Likert scale of 7, how much they felt that mathematics and programming were for them. The results show that the more participants feel like mathematics is for them, the less they use troubleshooters. With programming, it is the other way around. Furthermore, the participants, who found the weekly tasks and tests harder, used more troubleshooters. The users, who made more attempts to submit weekly tasks and tests and were deducted points for that, used troubleshooters more. This could indicate that the people falling behind do use the opportunity to use troubleshooters more.

Table 2

Spearman Correlation Coefficients Between Participants Evaluations on Various Statements and Their Evaluations on the Usage of Troubleshooters

Statement	**Evaluations**
Evaluation that mathematics is something for me	-0.128
Evaluation on programming pleasantness	0.261
Evaluation on the difficulty of last exercises	0.348
Evaluation on the difficulty of the last week's quiz	0.174
Number of attempts to submit solutions of exercises	0.300
Number of attempts to submit weekly quiz (at least 90% right solutions)	0.146
Sum of points of weekly quizzes	-0.223

**Note*. All coefficients are statistically significant on .01 level.

Discussion and Conclusions

Creating troubleshooters for course tasks has been useful, as the number of questions asked from the helpdesk declined 29%. In total, 86.5% of the users of troubleshooters have given at least 4 points from 7, showing agreement with the statement that troubleshooters were helpful. It is obvious that not all participants need troubleshooters, but troubleshooters as one possibility to replace a helpdesk could influence the attitude towards the MOOC (Warren et al., 2014) and could be one reason why in our MOOC the dropout rate was lower than in most MOOCs (Jordan, 2014; Siemens, 2013). As a result of the success of troubleshooters, they were also implemented in a MOOC, called *Introduction to Programming*, and will be used in the future.

This paper does not describe didactically how much troubleshooters can actually help in certain situations. The course *About Programming* uses only shorter basic tasks to evaluate certain aspects of the topics and the tasks have mostly one solution; however, the construction of troubleshooters can become very long and difficult in bigger tasks and algorithms. Tasks like finding suitable algorithmic solutions can form several branches, which make the troubleshooter's decision tree difficult to navigate. How much one can help with hints, when there are several different solutions, has not been looked at in this case. Pieterse (2013) stated that providing high quality automatic assessment can be very challenging and demands increased effort from the instructor. We think that the same applies to troubleshooters; however, crafting troubleshooters can be rewarding to the instructors as there is much to learn about learners' mistakes and problems (Vihavainen et al., 2012).

While the number of questions asked from the helpdesk has declined, many of the questions asked from the helpdesk duplicate the questions solved by troubleshooters. It still remains unknown why that occurs. There is a future course coming up without the helpdesk, which may lead to more answers.

Finally, troubleshooters change the way people study as, for example, many learners look at troubleshooters even before they encounter any problems, solve the tasks, or even before reading the theoretical materials about the topic. As has been suggested in a previous study (Anderson, Huttenlocher, Kleinberg, & Leskovec, 2014), learners differ in the ways they engage with online courses. Some participants acquire the required knowledge without needing troubleshooters, while other participants (called "solvers" by Anderson's et al., 2014) focus on solving exercises, using troubleshooters if they encounter problems. The survey revealed that 19.8% of the participants did not use troubleshooters at all and the results indicate that the participants, who received more points and felt that the weekly tasks were easier, were not very active in using troubleshooters, which could imply that troubleshooters are more helpful to people in need for extra assistance. In our case this tool was created for learning as was suggested by Admiraal et al. (2015) and therefore could be helpful for learners.

A danger is that troubleshooters can become an essential part of the study process, which can lead to learned helplessness, where some of the students are addicted to troubleshooters without even experiencing any problems. Will the students learn to swim when they have been thrown a swim ring?

As troubleshooters help to understand the content of the task, they can reduce the students' ability to read and understand the text of the task by themselves. Understanding the problem without external assistance, being able to solve a problem without hints, and debugging it by finding the solutions yourself are important parts of programming too. Similar concerns were also highlighted in a previous study (Vihavainen et al., 2012).

Creating troubleshooters requires special kind of experience and is not that easy. In MOOCs, each new task has to i) use mostly the knowledge taught before, ii) have an automated assessment feedback, and iii) have a troubleshooter with hints for the questions that may occur. All this limits the creation and changing of tasks because too many changes would have to be made. Development of the *Murelahendaja* environment continues in further courses.

Acknowledgments

We would like to thank all of the participants and organizers of the MOOC for their co-operation. Also, the support of the University of Tartu for the development of the MOOC, data collection, and writing of the paper has been considerable.

References

Admiraal, W., Huisman, B., & Pilli, O. (2015). Assessment in massive open online courses. *Electronic Journal of e-Learning*, *13*(4), 207-216.

Anderson, A., Huttenlocher, D., Kleinberg, J., & Leskovec. J. (2014). Engaging with massive online courses. In *Proceedings of the 23rd International Conference on World Wide Web* (WWW '14), 687-698. ACM, New York, NY, USA. doi:10.1145/2566486.2568042

Baturay, M. H. (2015). An overview of the world of MOOCs. *Procedia - Social and Behavioral Sciences*, *174*, 427-433. doi:10.1016/j.sbspro.2015.01.685

Downes, S. (2017). New models of open and distributed learning. In M. Jemni, Kinshuk, & M. Khribi (Eds.), *Lecture notes in educational technology. Open education: from OERs to MOOCs* (pp. 1-22). Berlin, Germany: Springer-Verlag. doi:10.1007/978-3-662-52925-6_1

Ebben, M., & Murphy, J. S. (2014). Unpacking MOOC scholarly discourse: A review of nascent MOOC scholarship. *Learning, Media and Technology*, *39*(3), 328-345. doi:10.1080/17439884.2013.878352

Garner, S., Haden, P., & Robins, A. (2005). My program is correct but it doesn't run: A preliminary investigation of novice programmers' problems. In *Proceedings of the 7th Australasian Conference on Computing Education*, 173-180. Retrieved from http://crpit.scem.westernsydney.edu.au/confpapers/CRPITV42Garner.pdf

Greene, J. A., Oswald, C. A., & Pomerantz, J. (2015). Predictors of retention and achievement in a massive open online course. *American Educational Research Journal*, *52*(5), 925-955. doi:10.3102/0002831215584621

Hanks, B. (2008). Problems encountered by novice pair programmers. *Journal on Educational Resources in Computing, 7*(4). doi:1316450.1316452

Hew, K. F., & Cheung, W. S. (2014). Students' and instructors' use of massive open online courses (MOOCs): Motivations and challenges. *Educational Research Review*, *12*, 45–58.

Hone, K. S., & El Said, G. R. (2016). Exploring the factors affecting MOOC retention: A survey study. *Computers & Education*, *98*, 157-168. doi:10.1016/j.compedu.2016.03.016

Jordan, K. (2014). Initial trends in enrolment and completion of massive open online courses. *The International Review of Research in Open and Distributed Learning*, *15*(1). doi:10.19173/irrodl.v15i1.1651

Lepp, M., Luik, P., Palts, T., Papli, K., Suviste, R., Säde, M., & Tõnisson, E. (2017a). MOOC in programming: A success story. In *Proceedings of the International Conference on e-Learning* (ICEL), 138-147. Academic Publishing International, USA.

Lepp, M., Luik, P., Palts, T., Papli, K., Suviste, R., Säde, M., Hollo, K., Vaherpuu, V., & Tõnisson, E. (2017b). Self- and automated assessment in programming MOOCs. In D. Joosten-ten Brinke, & M. Laanpere (Eds.), *Communications in computer and information science: Vol. 653. Technology enhanced assessment* (pp. 72-85). Cham, Switzerland: Springer International Publishing AG. doi:10.1007/978-3-319-57744-9_7

Marceau, G., Fisler, K., & Krishnamurthi, S. (2011). Measuring the effectiveness of error messages designed for novice programmers. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (SIGCSE '11), 499-504. ACM, New York, NY, USA. doi:10.1145/1953163.1953308

McRitchie, K. J. (2009). Technology consultants: A successful generation of student staff. In *Proceedings of the 37th Annual ACM SIGUCCS Fall Conference: Communication and Collaboration* (SIGUCCS '09), 287-292. ACM, New York, NY, USA. doi:10.1145/1629501.1629555

Motsuk, K. (1999). The student helpdesk: student information technology support at Georgetown University. In *Proceedings of the 27th Annual ACM SIGUCCS Conference on User Services: Mile High Expectations* (SIGUCCS '99), 172-175. ACM, New York, NY, USA. doi:10.1145/337043.337135

Muuli, E., Papli, K., Tõnisson, E., Lepp, M., Palts, T., Suviste, R., Säde, M., & Luik, P. (2017). Automatic assessment of programming assignments using image recognition. In É. Lavoué, H. Drachsler, K. Verbert, J. Broisin, & M. Pérez-Sanagustín (Eds.), *Lecture notes in computer science: Vol. 10474. Data driven approaches in digital education* (pp. 153-163). Cham, Switzerland: Springer International Publishing AG. doi:10.1007/978-3-319-66610-5_12

Onah, D. F. O., Sinclair, J. E., & Boyatt, R. (2014). Exploring the use of MOOC discussion forums. In *Proceedings of London International Conference on Education*, 1-4. doi:10.13140/RG.2.1.3319.5042

Papathoma, T., Blake, C., Clow, D., & Scanlon, E. (2015). Investigating learners' views of assessment types in massive open online courses (MOOCs). In G. Conole, T. Klobučar, C. Rensing, J. Konert, & E. Lavoué (Eds.), *Lecture notes in computer science: Vol. 9307. Design for teaching and learning in a networked world* (pp. 617-621). Cham, Switzerland: Springer International Publishing AG. doi:10.1007/978-3-319-24258-3_72

Pieterse, V. (2013). Automated assessment of programming assignments. In *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research* (CSERC '13), 45-56. Open Universiteit, Heerlen, The Netherlands. Retrieved from http://dl.acm.org/citation.cfm?id=2541917.2541921

Sanders, K., & Thomas, L. (2007). Checklists for grading object-oriented CS1 programs: Concepts and misconceptions. *SIGCSE Bull*, *39*(3), 166-170. doi:10.1145/1269900.1268834

Siemens, G. (2013). Massive open online courses: Innovation in education. *Open educational Resources: Innovation, Research and Practice*, *5*. Retrieved from https://oerknowledgecloud.org/sites/oerknowledgecloud.org/files/pub_PS_OER-IRP_web.pdf#page=31

Sinnett, C. J., & Barr, T. (2004). OSU helpdesk: A cost-effective helpdesk solution for everyone. In *Proceedings of the 32nd annual ACM SIGUCCS conference on User services* (SIGUCCS '04), 209-216. ACM, New York, NY, USA. doi:10.1145/1027802.1027851

Skaanning C., Jensen F.V., & Kjærulff U. (2000). Printer troubleshooting using bayesian networks. In R. Logananthara, G. Palm, & M. Ali (Eds.), *Lecture notes in computer science: Vol. 1821. Intelligent problem solving. Methodologies and approaches* (pp. 367-380). Berlin, Germany: Springer-Verlag. doi:10.1007/3-540-45049-1_45

Suen, H. K. (2014). Peer assessment for massive open online courses (MOOCs). *The International Review of Research in Open and Distance Learning, 15*(3), 313-327. doi:10.19173/irrodl.v15i3.1680

Veletsianos, G., & Shepherdson, P. (2016). A systematic analysis and synthesis of the empirical mooc literature published in 2013–2015. *International Review of Research in Open and Distributed Learning*, *17*(2). doi:10.19173/irrodl.v17i2.2448

Vihavainen, A., Luukkainen, M., & Kurhila, J. (2012). Multi-faceted support for MOOC in programming. In *Proceedings of the 13th Annual Conference on Information Technology Education*, 171-176. ACM. doi:10.1145/2380552.2380603

Warren, J., Rixner, S., Greiner, J., & Wong, S. (2014). Facilitating human interaction in an online programming course. In: *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (SIGCSE '14), 665-670. ACM, New York, NY, USA. doi:10.1145/2538862.2538893

Washburn, K., & El-Bayoumi, J. (2003). UNB's single site service source. In *Proceedings of the 31st Annual ACM SIGUCCS Fall Conference* (SIGUCCS '03), 237-239. doi:http://dx.doi.org/10.1145/947469.947532

Wustner, S., Joumblatt, D., Teixeira, R., & Chandrashekar, J. (2012). Automated home network troubleshooting with device collaboration. In *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop* (CoNEXT Student '12), 61-62. ACM, New York, NY, USA. doi:10.1145/2413247.2413284