Increasing the consistency of feedback on programming code by using a tagging system

L.D. Steenmeijer
University of Twente
The Netherlands
l.d.steenmeijer@student.utwente.nl

ABSTRACT

Programming knowledge is a skill which has gotten more important and widely used over the years. So teaching this in the best and most efficient way is also becoming increasingly more relevant. Since giving feedback is an influential part of this process it is important that concise and consistent feedback is given.
The Atelier tool is created to easily give feedback on code in a social media like style. For this research it is extended with a tagging functionality. A focus session is conducted to look into tagging behaviour on programming code by TA's. In addition to this a usage test was also performed to test the tag implementation. At the end of the paper recommendation for future research will be made.

Keywords

Programming, Errors, Learning, Feedback, Atelier

1. INTRODUCTION

The proposed research will look into increasing the consistency of giving feedback on code by using tags to identify the programming errors. It will also look at which tag recommendation system is the best to use and in which categories programming mistakes can be organised.

To achieve this Atelier will be used. Atelier is a tool created for the Creative Technology program to support discussions and feedback between the students, lecturers and teaching assistants (TA) [7]. The tool will be used to give feedback and eventually tag the code.

Over the course of a computer science study, students often learn programming by doing it themselves. During this they can be positively influenced by receiving feedback [9, 13, 11]. Because of an increase of students in study programs where programming skills are required [4], there will be an increased amount of TA's, lecturers and others who will examine the code and give feedback. Since there is a wide variety of possible programming errors and interpretations thereof, this feedback given on the same code can widely differ [14]. The proposed research will look into increasing this consistency by using tags and supporting mechanisms to identify different programming problems.

33rd Twente Student Conference on IT Jul. 3rd, 2020, Enschede, The Netherlands.

1.1 Research Question

The question that will be researched is formulated as follows:

RQ1: Can feedback on programming code given by teaching assistants be increased in consistency by using a tagging mechanism to identify the programming error?

RQ1.1: What tagging mechanism is best suited for giving feedback on code in the Atelier environment?

RQ1.2: How can you embed tagging in feedback sessions of programming classes?

RQ1.3: Does tagging help in increasing the consistency of feedback given by TA's?

From the start of the project it was expected that the data that could be gathered would be at most from one week of collection, because of the time span of the research. This was never expected to provide enough data to serve as a basis for a scientific conclusion. Thus the focus of this research was mainly pointed at creating data collection methods so that this research could be more easily performed in the first module of next year and to provide recommendations for future research.

2. BACKGROUND

2.1 Atelier

Atelier is a tool created for first-year students of the Creative technology (CreaTe) program. It helps the teaching staff in giving feedback on code made by the students in the `Processing` language. Since programming skills in the CreaTe study are taught based on broad exercises with a lot of freedom for coming up with your own solution, the student can choose how they want to solve the assignment and thus the solutions differ widely. Therefore, the feedback given to the students can not be tested easily and a more individualised system was needed. This is where Atelier comes in. Atelier offers an online platform where the teaching staff can comment on and discuss the code in a style inspired by social media.[6]

2.1.1 Processing

Processing is a language based of Java created to be easily learned by novice programmers [12]. It is also focused on visual arts or a visual context. This makes it rewarding for new programmers since it will produce visual results early in the learning process. It also serves as introduction for the start of learning more advanced languages such as Java or C++ [8]. Processing can be seen as a so called "software sketchbook" since it is easy to implement and refine new ideas quickly [12].

2.2 Tags

A tag is a mechanism to connect metadata to a certain

object. In this case the metadata will be the type of programming mistake and the object the code in which this occurred.

Tag recommendation can be used to enhance the user's experience of using tags. By suggesting certain tags it minimises the chance of them having misspellings or not being relevant to the mistake.

There are two types of tag recommendation based on which target they refer to: object-centered and personalised. At the former the tags are only ranked based on their relation to the object, whereas the latter also takes the user who does the tagging into account. [2, 3]

Other aspects of a tag recommendation system are the objective on which the tags are recommended, these can either be relevance, diversity or novelty. [3]

The tag recommendation exists of two processes, the first is generating the set of tags and the second is ranking the tags. An example of a ranking techniques is based on tag co-occurrences, this will look at pairs of possible tags and the one chosen for a certain object. Based on this information a ranking can be made which tag will be chosen more often. This method does need a training set of data in which it is shown which tags were previously chosen and for which object. [2]

2.3 Programming mistakes

There is a wide variety of programming mistakes which means a clear but meaningful categorisation of these is essential for the tagging functionality to work appropriately. Programming mistakes can be split up in 3 main categories namely: syntax errors, semantic errors and type errors.[1] These are each very broad and not very informative so they need to be split up in more specific categories. Since educators find it difficult to point out correctly what mistakes are most frequently made by students [5], the categorisation system as developed by McCall in "A new look at novice programmer errors." [10] will be used as a partial basis for the initial set of the tag recommendation. These categories also include the most frequent mistakes as found by Altamadri [1].

The categories occurring in the top 20 of most severe errors will be used to form starting set of possible tags. These will cover 80% of the errors existing in the code, as stated by McCall [10]. These can be found in appendix A.

3. METHODOLOGY

3.1 Research question 1.1

3.1.1 Literature research

To answer the question "What tagging mechanism is best suited for giving feedback on code in the Atelier environment?" a literature research will be performed. To achieve this different papers will be looked at and read through for information. The research will look into keywords such as tags, tagging mechanism and tag recommendation.

3.2 Research question 1.2

3.2.1 Literature research

At first a preliminary literature research was performed to look into categories of tags which would be suitable for a tagging mechanism which gives feedback on code. The research will mostly look into different categories of programming problems to use as a basis for the recommended tags in the tagging mechanism.

3.2.2 Focus session

A focus session was chosen as part of the design research methodology used to create the tagging mechanism. The session will be performed to get a first look at how TA's give feedback and use tags. The focus session will consist of 4 groups of 2 TA's who comment on 4 different programs in 4 rounds. The tool that will be used for this session is Google docs, which everyone has used before and thus the tool itself will not be tested. Each group will get the program as a Google doc file and can use the comment function to give feedback. In the first round each group will get an uncommented program and instructions for giving feedback on this. The first two groups will only receive instructions to comment on the program, the third group will get instructed to use tags and the last group will also be given a list of recommended tags as well as the instructions to use tags.

In the second round the program will have Zita comments as well, these are already added to the comments. Zita is a plugin which is used in Atelier and it will automatically generate feedback on the style of the code. The Zita comments will also have tags added to them. In the third round each group will receive a piece of code with comments from another group. Finally, in the last round, the groups will get a program with comments from another group and Zita comments. At end of this the TA's will be asked to fill in a questionnaire and to participate in a discussion about the session.

This focus session will eventually result in outcomes about which tags are used, minutes about discussion and the questionnaires.

3.3 Research question 1.3

To answer this question a design research will be performed in which the the atelier extension with tagging functionality will be designed, tested and eventually be used in a longitudinal study. This will result in one implementation cycle with accompanying usage test.

3.3.1 Usage test

A usage test will be performed. The atelier extension will be designed and implemented according to the results as found in RQ 1.2. This will then be tested during a tutorial session in which the TA's will be instructed to use the tags while giving feedback. This will result in findings about the extension, whether it performed as expected or caused problems.

3.3.2 Longitudinal study

This study will not yet be performed during this research since the time span of it is too short. But eventually a longitudinal study needs to be done to research the answer to RQ 1.3. This will be explained in more detail in section 5.

4. RESULTS

4.1 Research question 1.1

To answer RQ 1.1 a decision for the tagging mechanism was made based on the available data. There was no previous data available on tags in Atelier, because tags were not as of yet used in Atelier. So the tagging mechanism could not be one based on already existing data about which tags were more often chosen in combination with objects the tags are linked to. Thus it was decided to create a tag ranking based on the 10 most used tags. This resulted in a recommendation from which the TA could draw inspiration and hopefully the most used tags will also be

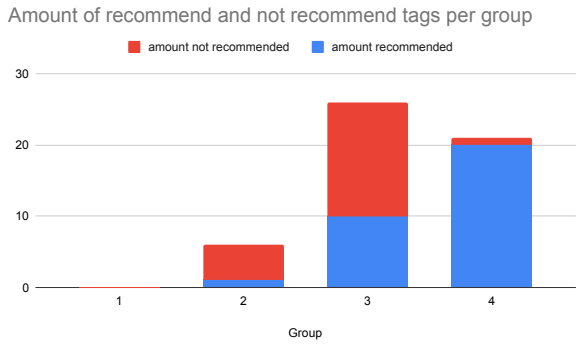

Figure 1. Chart showing groups and recommend vs not recommended relation

considered the most useful ones.

4.2 Research question 1.2

4.2.1 Literature research

The information gathered from the literature research performed for this question can be found in section 2.3.

4.2.2 Focus session

The data regarding the use of tags was found during the focus session this resulted in the following. All comments were looked at and the used tags counted and written down. These results can be found in appendix B. The tags that are marked green are the tags that were given as a recommendation, these were on average used 5,17 times whereas the ones without recommendation were used 1,84 times. Thus recommending tags will cause them to be used more often. When looking at Figure 1 it can also be seen that giving a group a certain list of tag recommendation will cause them to be less likely to come up with tag themselves. These tags are generally less technical and more descriptive about the problem. Another effect that can be seen is that the TA's who got a commented code file with tags are more inclined to use tags as well, compared to the TA's who got a file without tags. This can also be seen in Figure 1.

At the end of the focus session the TA's were also questioned about their ideas on the usage of tags. Some think that the use of tags alone would not really contribute anything. But if the tags would have added functionality they would be useful. A few suggestions they gave for these were the option to filter comments on which tags occur in them. It would also be useful if students could see which of the tags are most used in the comments on their code, this could be useful in determining in which areas they have the most problems. Another idea was to link the tags to a reference page with some examples, solutions and explanation about the tagged problem. The TA's also noted that it would be helpful to have tags in the same categories in which the exams will be graded. From this it could also be concluded that the objective of the tag recommendation is relevance, since the diversity or novelty of the tag is not important for giving feedback.

4.3 Research question 1.3

4.3.1 Extension of Atelier

Firstly the Atelier environment needs to be extended with the functionality to add tags to certain pieces of code. Since the Atelier project is developed with Typescript, React and Nodes.js this will also be the case for the new

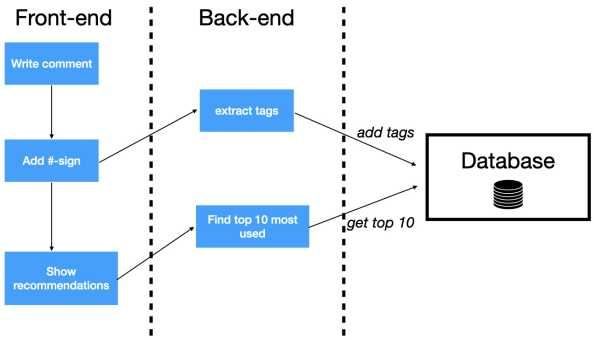

Figure 2. Overview of the extension of Atelier

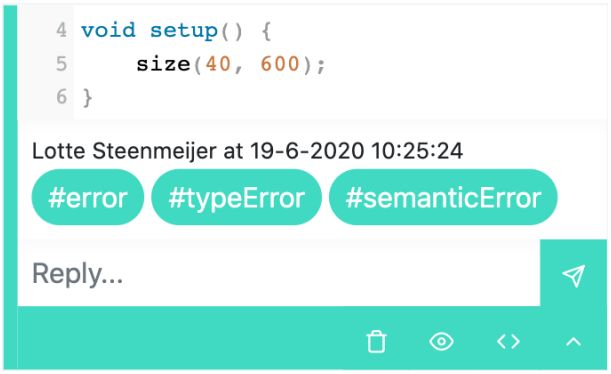

Figure 3. Screenshot of tagged code in Atelier

tagging functionality.

The tagging functionality entails the possibility to add a tag in a comment this is done by using the #-sign. The comment is sent to the back-end of Atelier wherein the tags are extracted from the comment text. These are then added to the Atelier database in a separate tag database with columns for id of the tag, the id of the comment and the body of the tag. This table can then be used to return the most used tags in the tag recommendation. This is implemented at the front-end of atelier, when a comment is created and a tag is started the recommendations will be shown. An overview of this process can also be seen in Figure 2. In Figure 4 and 3 two screenshots can be seen of the implemented tagging mechanism.

4.3.2 Usage test

A usage test was performed in the form a tutorial in which this extension was used. During the tutorial a few TA's used the tags and no errors occurred. In Figure 4 the top 10 most used tags are shown after the usage test, this will evolve more as the tagging functionality is used more often.

5. FUTURE RESEARCH

Since most of the data collection will happen during a longitudinal study in the future, no conclusion can yet be drawn from this. However future research recommendation can be made these will be described in this section. First a recommendation for a study about consistency which can be performed during module 1. Then studies about the most frequently used tags, influence of tags on test results, helpfulness of feedback and overall use of tags. Finally some recommendation about improvements of the tag implementation. The remainder of this section will give more details on each of these proposed research

Figure 4. Screenshot of tagging code in Atelier.

ideas.

5.1 Consistency research in module 1

A good opportunity for future research will be the first module of the CreaTe program. In this module Atelier will again be used as a feedback tool. Since the data collection could already start taking place in the first week of the module a lot more data could be gathered. During the first five weeks of this module a sufficiently large group TA's could be asked to give feedback in two groups. One of these group will be asked to provide feedback on the code of students with comments and without the use of tags. Whereas the other group will receive the same code and instructions to comment on it and to use tags in the comments as well. This will result in two different sets of comments: one with tags and the other one without. For each of these sets the consistency can be calculated and compared to the consistency of the other group.

5.2 Most frequently used tags

More analysis is now made possible by the tag extension of Atelier. Because of this more data on the usage of tags during feedback sessions can be gathered. Other future analysis that could be performed is an analysis of which tags are most frequently used and in which categories these belong. The focus session hinted at the use of more descriptive tags. These indications could be more thoroughly analysed by using the data from atelier.

5.3 Influence of tags on test results

The use of tags to indicate the nature of the programming problem could be used to analyse which students make which mistakes the most. This could then be linked to performance of the students on tests and it could be researched whether the amount of tagged problems gives an indication of the student's performance on the test. This could then be used to increase the support for some students in advance of the test to pass it.

5.4 Helpfulness of feedback

In addition to these analyses another one could be made, when Atelier would be extended with an additional feature in which the students could rate the feedback they are given. This could be done with a simple one-up-until-five-star-system or a thumbs-up thumbs-down mechanism. This data could then be combined to research whether the use of tags in a comment makes the feedback more helpful for the student. Another aspect which could be looked at is whether TA's who create more new tags also give more useful feedback.

5.5 Overall use of tags

One more analysis that could be performed is to look at whether tags are used at all. This could answer the question whether TA's would actually use the new tagging functionality. This could be done with a combination of data about the total amount of comments made and the total amount of tags used. It could then be calculated what the average amount of tags in a comment is and how much comments are made without tags against comments made with tags.

5.6 Improvements of tag implementation

The tag implementation in Atelier can also be more improved. The suggestions of the TA's as mentioned in section 3.2.2 can be implemented. Another improvement which can be made in the future is the implementation of another recommendation mechanism. When sufficient training data is gathered with Atelier the recommendation can be made more specific, one example would be to personalise the recommendations. This way the TA's will receive tag recommendations based on which tags they use the most. This could also be combined with object centered recommendations, which would make a recommendation based on which tags are more often used in a specific course or exercise. Another addition could be the inclusion of certain criteria on which the exams are also graded. A teacher could add these to a certain exercise or course and tags for these criteria would be higher in the recommendation ranking.

6. CONCLUSION

The research conducted looked into the use of tags while giving feedback and whether this increases the consistency of the feedback. This also included looking into the best tag recommendation mechanism, which eventually ended up being a mechanism based on the top 10 most used tags. The tagging mechanism was implemented in the Atelier tool and tested with a user test. A focus test was also performed to look into the use of tags by TA's while giving feedback. From this it could be seen that TA's will use tags that are recommended more. As well as an indication that when the commented code already has tags used in it that the TA's will emulate this behaviour.

And the end of the paper recommendations for future long-term studies were given. These will look into the consistency of feedback during a longitudinal study, the most fre-

quently used tags, the influence of tags on test results, the helpfulness of feedback, the overall use of tags and on improvements of the tag implementation. The implemented tag extension serves as a basis for these recommendations and will make it easier to execute them.

7. REFERENCES

[1] A. Altadmri and N. C. Brown. 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *SIGCSE 2015 - Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 522–527, New York, New York, USA, 2 2015. Association for Computing Machinery, Inc.

[2] F. Belém, J. Almeida, and M. Gonçalves. Tagging and Tag Recommendation. In *Text Mining - Analysis, Programming and Application [Working Title]*, pages 1–14. IntechOpen, 9 2019.

[3] F. M. Belém, J. M. Almeida, and M. A. Gonçalves. A survey on tag recommendation methods. *Journal of the Association for Information Science and Technology*, 68(4):830–844, 4 2017.

[4] B. Bizot, E. Walker, and H. College. Generation CS: Computer Science Undergraduate Enrollments Surge Since. Technical report, 2017.

[5] N. C. Brown and A. Altadmri. Novice Java programming mistakes: Large-scale data vs. educator beliefs. *ACM Transactions on Computing Education*, 17(2):1–21, 5 2017.

[6] A. Fehnker and A. Mader. Atelier for Creative Programming Stimuleringsregeling Open en Online Onderwijs. pages 1–13, 2019.

[7] A. Fehnker and A. H. Mader. Atelier for Creative Programming. `https://www.utwente.nl/en/eemcs/fmt/research/projects/atelier/#summary`.

[8] B. Fry and C. Reas. Processing.org. `https://processing.org/`.

[9] J. Hattie and H. Timperley. The Power of Feedback. *Review of Educational Research*, 77(1):81–112, 3 2007.

[10] D. McCall and M. Kölling. A new look at novice programmer errors. *ACM Transactions on Computing Education*, 19(4):1–30, 7 2019.

[11] Y. Qian and J. D. Lehman. Using Targeted Feedback to Address Common Student Misconceptions in Introductory Programming: A Data-Driven Approach. *SAGE Open*, 9(4), 2019.

[12] C. Reas and B. Fry. Processing: Programming for the media arts. *AI and Society*, 20(4):526–538, 9 2006.

[13] V. J. Shute. Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189, 3 2008.

[14] S. H. S. Wong and A. J. Beaumont. A quest for helpful feedback to programming coursework. *Engineering Education*, 7(2):51–62, 2012.

APPENDIX

A. PROGRAMMING MISTAKES

Category	Description
variable not declared	the variable is called but was never declared
incorrect variable declaration	the variable was incorrectly declared
syntax error	e.g. confusing assignment operators, unbalanced parentheses
incorrect method declaration	method was incorrectly declared
semantic error	e.g. using == instead of .equals
type error	invoking methods with the wrong arguments
variable name written incorrectly	a spelling mistake was made in the variable name
missing return statement	the return statement in a non-void method is missing or can not be reached
method call targeting wrong type	method is called with the wrong type
method not declared	the method was never declared
incorrect attempt to use variable	the variable was incorrectly used
; missing	; is missing
method name written incorrectly	spelling mistake was made in the method name
class or type name written incorrectly	spelling mistake in the class or type name
parameter number mismatch	wrong number of parameters
use of non-static method from static context	a non-static method is used as if it were static
incorrect method call	method was called incorrectly
parameter type mismatch	the type of the parameter was incorrect
type mismatch in assignment	the type in the assignments do not match
extraneous closing curly brace	too much curly braces

B. FOCUS SESSION RESULTS

tags	Group 1			Group 2				Group 3				Group 4				
	B0A	D2A	E3B	F7C	D2A	F7C	E9A	B0C	E9A	B0C	F7A	D2C	F7A	D2C	B0A	E9B	total
new comments	18	2	16	5	18	19	2	12	11	7	8	2	12	6	5	4	
classes	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
compliment	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
confused	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
confusing	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	3
conventions	0	0	0	0	0	0	1	0	2	1	1	0	0	0	0	0	5
don't	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	2
eventhandling	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	2
functionality	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
hard coded	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
images	0	0	0	0	0	0	1	0	2	0	0	0	0	0	0	0	3
layout	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	3
loops	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	2
namingConventions	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
semantics	0	0	0	0	0	0	0	0	1	1	0	0	2	0	0	0	4
structure	0	0	0	0	0	0	0	0	1	2	1	1	1	2	0	0	8
style	0	0	0	0	0	0	1	0	0	0	1	0	5	3	2	1	13
typo	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
wrong	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1