10.20378/irb-56092

An Investigation into Freeform, Dynamic, Digital Annotation for Understanding Program Code

Craig Sutherland, Andrew Luxton-Reilly, Beryl Plimmer
Department of Computer Science
University of Auckland
New Zealand
{cj.sutherland|a.luxton-reilly|b.plimmer}@auckland.ac.nz

Abstract. Understanding program code is a time-consuming task. Previous research has found freeform annotations to be an invaluable tool in assisting comprehension for other forms of prose. Therefore freeform annotations may also increase code comprehension. However code has some unique characteristics compared to other forms of prose. My research is investigating how we can implement freeform annotations inside a code editor in a way that will reduce the time programmers spend on understanding code.

1 Research Area and Topic

My research area is stylus input computing and digital ink annotations.

I am investigating how freeform, digital ink can assist programmers comprehend computer program code and how freeform annotations should automatically adapt when the context changes.

2 Research Problem

Programmers spend a lot of time understanding computer program code. Program code differs from other forms of prose: it is non-linear, highly structured and dynamic. These differences increase the cognitive demands [1].

Programmers need to remember complex paths through the program code. This requires remembering what they have previously read and being able to quickly navigate through the code. Indeed, integrated development environments provide many tools that implement these two tasks. But these tools are infrequently utilised; partly due to the learning curve required [2].

Annotations assist comprehension by reducing the cognitive workload. They allow the reader to offload information from their short term memory and streamline navigation through a document [3]. This suggests annotations

could help reduce the effort required by programmers when reading program code.

The problem with current program annotation tools is they are text-based: forcing a reader to type text increases their cognitive workload [4]. In contrast, hand writing annotations does not appear to increase cognitive workload [4]. Therefore, if a programmer could use hand-written annotations on their code they would realise the benefits of annotations. However very few programmers print computer code because its size and dynamic nature. Instead programmers read code on-screen which limits their ability to hand-write annotations.

There are now tools that allow programmers to annotate code on-screen within their development environment (e.g. [1]). These tools proved that it is possible to annotate code on-screen but they also identified several other challenges.

One challenge is how to maintain the meaning of annotations. Programmers often want to understand code they are changing. If they annotate some code and then change the code then the annotations lose their meaning. Ideally annotations should adapt to changes in the content but people are unused to seeing their annotations change. Research is needed to identify when and how annotations should adapt in response to changes in the underlying content.

A second challenge is how to integrate annotations in a way that enhances comprehension. Previous research identified benefits of freeform annotations in digital readers for comprehension [5]. Further investigation is needed to see if these benefits also apply to program code.

3 Research Claim

Freeform, dynamic, digital ink annotations will reduce the time programmers need when trying to understand computer code.

4 Methods

The first phase of my research was a user study investigating how programmers annotate program code on paper. I observed experienced programmers reading and annotating code and then interviewed them asking why they annotated and the significance of their annotations. These findings have helped formulate the requirements of an annotation system for programmers. This study is to be reported at INTERACT 2015 [6].

The second phase is testing different ways of automatically changing annotations when the underlying code changes. The literature review identified some ways that annotations could be modified but these are very limited. In this study the participants read and annotate some code in an extension to Visual Studio. When they finish I modify the code underneath their annotations so the annotations are adapted. The participants rank the different automatic adaption routines based on preserving the original meaning. This is followed by an interview on what they think of the different adaptions.

The final phase will use the results from the previous phases to develop and evaluate an annotation system for programmers. The first phase provided the background on programmers' annotation practices. These will be used for building the workflow to support annotation. The second phase will provide an implementation of dynamic annotations. The aim of the final phase is a system that provides complete end-to-end support that assists program code comprehension. This solution will be validated in a field trial with software engineering students.

5 Proposed Solution

The final proposed solution will contain two parts: algorithms for automatically adapting annotations in response to code changes; and tools to assist programmers in navigating the code.

The algorithms use the type of annotation and the underlying code so the original meaning of the annotations is preserved.

The navigation tools will assist the programmer in both re-finding previous code and serendipitously finding new code. These will work without increasing the cognitive workload of the programmer.

6 Expected Contributions

A systematic literature review on how digital ink annotations have been implemented previously.

- Some guidelines on how to automatically adapt annotations in response to context changes.
- Formal evaluations of:
  - User expectations around how users react to automatic changes in their annotations and;
  - How annotations can support code comprehension.

7 Statement of Work

To date I have completed the systematic literature review. The review has been submitted to a journal for publication and we are waiting for the feedback. This study identified gaps around user expectations of automatic adaptations and how freeform annotations can be used to assist with software comprehension.

The first user study has been completed and accepted for publication at INTERACT 2015 [6].

The second user study has ethics approval and I have started collecting data. The prototype has been implemented allowing programmers to annotate code in Visual Studio. Annotations are automatically adapted using six different algorithms based on the type of annotation.

I have started implementing the functionality for the final iteration. This is based on requirements gathered from the first user study. However I have still to decide exactly which functionality to implement and how it should be tested.

My current issues are exactly what functionality would be beneficial for reading comprehension and how to assess the effectiveness of the functionality. I am thinking of doing two to three rounds of testing. After each round I will tweak the functionality to try and improve comprehension. These are the issues I would like to discuss with the consortium to gain some feedback on the final implementation.

References

[1] Priest, R. and Plimmer, B., "RCA: experiences with an IDE annotation tool," in Proceedings of the 7th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: design centered HCI, Christchurch, New Zealand, 2006, pp. 53-60.

[2] Maalej, W., Tiarks, R., Roehm, T., and Koschke, R., "On the Comprehension of Program Comprehension," ACM Trans. Softw. Eng. Methodol., vol. 23, pp. 1-37, 2014.

[3] Crisp, V. and Johnson, M., "The use of annotations in examination marking: opening a window into markers' minds," British Educational Research Journal, vol. 33, pp. 943-961, 2007.

[4] O'Hara, K. and Sellen, A., "A comparison of reading paper and on-line documents," in Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, Atlanta, Georgia, USA, 1997, pp. 335-342.

[5] Schilit, B. N., Golovchinsky, G., and Price, M. N., "Beyond paper: supporting active reading with free form digital ink annotations," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Los Angeles, California, USA, 1998, pp. 249-256.
[6] Sutherland, C. J., Luxton-Reilly, A., and Plimmer, B., "An Observational Study of How Experienced Programmers Annotate Program Code," in INTERACT 2015, Bambery, Germany, 2015.