Influence of Synchronized Domain Visualizations on Program Comprehension

Nuno Oliveira
University of Minho
Braga, Portugal
nunooliveira@di.uminho.pt

Maria João Varanda Pereira
Polythecnic Institute of Bragança
Bragança, Portugal
mjoao@ipb.pt

Daniela da Cruz
University of Minho
Braga, Portugal
danieladacruz@di.uminho.pt

Mario Berón
National University of San Luis
San Luis, Argentina
mberon@unsl.edu.ar

Abstract

An effective program comprehension is reached when it is possible to view and relate what happens when the program is executed, synchronized with its effects in the real world concepts. This enables the interconnection of program's meaning at both problem and program domains.

To sustain this statement we need (i) to develop a tool which provides and synchronizes views at both domains, and (ii) to perform an experiment to measure the actual impact of this approach.

So, in this working session we aim at discussing the benefits of providing synchronized domain visualizations. We also envisage to discuss the preparation and conduction of appropriate experiments that will test that benefits.

A case study will be used and the discussion will be supported by experimental material specially prepared for the occasion, but adapted from material already used in previous experiments.

1 Introduction

Brooks [1] and others [5, 6, 11], stated that a complete understanding of a program is reached when the analyst can relate the program domain—how statements are executed (operational semantics)—with the problem domain—what are the effects caused by the execution of those statements (logical semantics).

We adhered to such approach and started the systematic development of tools to help system maintainers to understand programs. The construction of Program Comprehension (PC) tools involves the use of specific methods and techniques to: (i) extract data from code, (ii) store the information gathered, and (iii) explore it to obtain new knowledge. However, this is not enough to attain the effectiveness of these tools, it is also required to research other areas, for example, human perception, reasoning and cognitive models. This is used to create specific artifacts for user interaction like graphic representations, navigation facilities, animations and other interface features.

Working with generic programming languages (GPLs), it was possible to develop generic tools to explore the source code at program level, but it could not be expected the same possibility concerning the visualization and manipulation at problem level. Motivated by the hypothesis that in a narrower and well defined domain we could synchronize the internal program execution and the effects provoked in the objects of the problem domain, we believe that we should focus on Domain Specific Languages (DSLs) to study Brooks theory.

To fire up the discussion around this topic, we will take as case study Alma^2 [9], an upgraded version of Alma [2], developed to cope with programs written in DSLs. Alma system only provides program domain visualizations based on program internal representations. Alma^2 takes a program, written in a DSL, and animates it (trough an abstract interpretation), displaying in a synchronized mode what happens internally during the execution of each statement and what are the effects of the execution over a graphical representation of the problem domain.

So, Alma^2 enabled us to study the behavior of domain specific programs, observing the synchronized visualizations of both domains; it will be used to test the working hypotheses.

The way we found to do such tests is to design experimental sessions on program understanding with Alma and Alma^2, measuring the user reaction in terms of time and accuracy of the answers.

In this working session we will take a set of already pre-

pared questionnaires related with our case study to discuss and reason about two main topics: (i) the *experiment preparation and execution*; (ii) the *effectiveness of that program comprehension approach*.

In section 2 we present motivation for the working session, based on previous work. In section 3 we describe the several parts of the session. In subsection 3.1 we enhance some of the topics that we would like to discuss about the preparation of the experimental session. In subsection 3.2 we detail the the final discussion on program comprehension approach that should be fired by the case study presented in the first part.

2 Motivation and Previous Work

Program comprehension approaches must be assessed in practice. In a first moment we may believe that materializing such an approach into a program comprehension tool (`PCTool`) is the complete way to assess a `PC` approach. However this does not provide a full assessment to the effectiveness of the approach. In the limit this is only a proof of concept. For a complete assessment we need to measure the influence of the tool (which embodies the `PC` approach) in the comprehension of programs. This means that we need to use real analysts to test not the usability of the tool, but the help it offers when they are through traditional comprehension tasks.

As was mentioned in the introduction, we conceived an approach for program comprehension stating that a program is understood when the analysts are able to connect the program and problem domain by means of synchronized visualizations of each one of these domains. Then, we developed a tool, `Alma`^{2}, that materializes the idea.

After `Alma`^{2} being developed, we prepared and guided an experiment with real users. In that experiment, there were involved 20 persons from computer sciences. This sample was a blend of undergraduate and postgraduate students. The undergraduate students were starting their final year of the *B.Sc.* degree; about 20% of the master students were starting their final year, and the remainder were finishing their master's thesis. All the *Ph.D.* students were starting their third year of studies. Their areas of specialization were varied and addressed fields like computer networks, distributed computing, bioinformatics, data warehousing, computer graphics and human-computer interaction.

With this experiment we gathered important numerical data to conclude about the influence of the approach in the execution of traditional program comprehension tasks. However, we found some flaws on the execution of the experiment that could compromise the final results. Moreover, we were not able to receive feedback about the techniques they used to build their mental models; therefore we were not able to reason about the relations between the approach and the cognitive theories.

In this context, we propose a working session to discuss about the experiment and the approach itself. Section 3 provides a more detailed explanation of the working session.

3 A Session with `Alma`^{2}

Briefing up, we plan to divide the working session into three parts:

1. Case study presentation
2. Discussion about the experiment
3. Discussion about the program comprehension approach

The last two points are clearly explained below.

3.1 Discussion about the experiment

The preparation and execution of the experiment followed the standardized guidelines provided by *Di Penta et al.* in [4].

To measure the influence of our approach in the comprehension of programs we must answer the following research question:

> *Does the synchronized visualization of the program and problem domains help users to understand programs?*

This research question led us to set the hypothesis **H1**. We also consider three sub-hypotheses that are directly related with **H1**, but are easier to measure with the answers provided by the participants in the questionnaire.

- **H0 (null hypothesis)** — Synchronized visualizations of problem and program domains, have no influence on program comprehension.
- **H1** — Synchronized visualizations of problem and program domains, really *have* influence on program comprehension.
- **H1.1** — Synchronized visualizations of problem and program domains have influence in the *time* spent on the comprehension tasks, decreasing it.
- **H1.2** — Synchronized visualizations of problem and program domains have influence in the *correctness* of the comprehension tasks, augmenting it;
- **H1.3** — The problem domain visualizations and their synchronization with the program domain visualizations are *important* to the process of comprehension;

To validate these hypotheses, we used a questionnaire based on two dimensions allowing us to verify: on the one hand, if the participants perceived the program functionalities; and on the other hand, if the participants were able to evolve the program.

These dimensions must be assessed both with `Alma` and `Alma`^{2}. The use of `Alma` would provide base results that will be compared with those coming from the use of `Alma`^{2}. The comparison of results will answer the research question. In this context, the tasks of the experiment not only address details on perceiving and evolving programs, but also are oriented to answer the research question, by asking directly the opinion of the participants about the influence of `Alma`^{2} in their performance. Moreover, each task of the questionnaire provides data to validate the three sub-hypotheses rose earlier in this section: (i) to gather information for **H1.1**, the participant is required to stamp the time at the begin and at the end of the task execution; (ii) to capture information for **H1.2**, the participants should answer questions about several programs, requiring a partial or full understanding of their purpose or behavior, and finally (iii) to gather information for **H1.3**, the participant is asked to score the importance of the problem domain visualization and its synchronization with the program domain visualization, for the resolution of the task. Although this is a subjective answer for each participant, the overall result reflects the general opinion.

The questionnaire is divided into two groups: one group for perceiving questions and another for evolving questions. We will be using two different `DSL`s: a well-known imperative language to control the Karel robot [8, 10], and a declarative language, Lavanda, which describes a laundry ordering list [3, 7]. *Perceiving* and *evolving questions* are associated to each language. Moreover, each one of these questions are to be answered, alternately, using `Alma` and `Alma`^{2}, and without any time restriction. This will allow the combination of two languages with two tools and two types of questions. Questions will require an open answer, a direct answer, or multiple choice answers. The adoption of this format, eases the correction of the questionnaire because the questions are either correct or wrong. The open answer questions add some difficulty on this matter, because there are multiple ways to achieve the correct solution, so, it requires further analysis on its correction.

In this part of the working session and after a brief presentation of the case study, we will discuss the experiment and the elaborated questionnaire. The realization of a controlled experiment is not yet standardized. Although there are works on that direction, like in [4], there are always experiments not following such standardization and go through *ad-hoc* means to achieve the same results. We tried to prepare a standard experiment, however in some cases we may have missed it and go through non-standard ways. So, the objective of this first discussion topic is to gather some feedback from the experts about what was good, what was bad, and what was ugly in the way we prepared and conducted the experimental session. The brain storming on this topic will generate important notes that are useful to all the participants that may want to prepare an experiment on program comprehension in the future.

3.2 Discussion about the Program Comprehension Approach

In this part of the working session, we will discuss and reason about the two other topics announced in the introduction:

1. the effectiveness of the approach and
2. the relation between this new approach and the several models on cognitive theory.

In the first topic the discussion will be about the approach for program comprehension based on the synchronization of program and problem domains. Experts on program comprehension studies are in a good position to discuss the benefits of our approach. We must remind that the approach is not the tool but the idea materialized by the tool. So, we are not evaluating `Alma`^{2} but the influence of its visualizations in the construction of mental models. Summing up, we are aiming at having feedback about how the approach was used to comprehend the programs.

Finally, in the second topic we will discuss and reason about how we can relate the approach with the several cognitive models in the cognitive theory, namely the bottom-up and top-down strategies. This discussion may be similar or tangled with the discussion of the previous topic, nevertheless, we are convinced that there are some straight relation between the approach and the adoption of a systematic cognitive strategy for mental model construction.

References

[1] R. Brooks. Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, 18(6):543–554, November 1983.

[2] D. da Cruz, P. R. Henriques, and M. J. V. Pereira. Constructing program animations using a pattern-based approach. *ComSIS – Computer Science an Information Systems Journal, Special Issue on Advances in Programming Languages*, 4(2):97–114, 2007.

[3] D. da Cruz, M. J. V. Pereira, M. Beron, R. Fonseca, and P. R. Henriques. Comparing generators for language-based tools. In *Proceedings of the 1.st Conference on Compiler Related Technologies and Applications, CoRTA'07 — Universidade da Beira Interior, Portugal*, July 2007.

[4] M. Di Penta, R. E. K. Stirewalt, and E. Kraemer. Designing your next empirical study on program comprehension. In *ICPC '07: Proceedings of the 15th IEEE International Conference on Program Comprehension*, pages 281–285, Washington, DC, USA, 2007. IEEE Computer Society.

[5] S. Letovsky and E. Soloway. Delocalized plans and program comprehension. *Software, IEEE*, 3(3):41–49, 1986.

[6] D. C. Littman, J. Pinto, S. Letovsky, and E. Soloway. Mental models and software maintenance. *J. Syst. Softw.*, 7(4):341–355, 1987.

[7] N. Oliveira. Improving program comprehension tools for domain specific languages. Master's thesis, University of Minho, Braga, Portugal, October 2009.

[8] N. Oliveira, P. R. Henriques, D. da Cruz, M. J. V. Pereira, M. Mernik, T. Kosar, and M. Črepinšek. Applying program comprehension techniques to karel robot programs. In *Proceedings of the International Multiconference on Computer Science and Information Technology - 2nd Workshop on Advances in Programming Languages (WAPL'2009)*, pages 697–704, Mragowo, Poland, October 2009. IEEE Computer Society Press.

[9] N. Oliveira, M. J. V. Pereira, P. R. Henriques, and D. da Cruz. Visualization of domain-specific program's behavior. In *Proceedings of VISSOFT 2009, 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 37–40, Edmonton, Alberta, Canada, September 2009. IEEE Computer Society.

[10] R. Pattis. *Karel, The Robot: A Gentle Introduction to the Art of Programming*. John Wiley and Sons, Inc., 1st edition, 1981.

[11] A. von Mayrhauser and A. M. Vans. Program understanding - a survey. Technical report, Colorado State University, August 1994.