What Will the User Do (Next) in the Tool?*

Darko Marinov
University of Illinois at Urbana-Champaign
marinov@cs.uiuc.edu

Sarfraz Khurshid
The University of Texas at Austin
khurshid@ece.utexas.edu

ABSTRACT

This position paper advocates the importance of analyzing how users interact with tools that aid software development. Answering "What will the user do (next) in the tool?" can help improve not only the tool's usability but also its underlying technology in a common usage scenario. While a lot of progress has been made in improving the underlying technology for such tools, less effort has been spent on studying how the tools are typically used in practice.

We present a summary of our previous, initial study on how (beginner) users interact with the Alloy Analyzer, a tool for automatic analysis of software models written in Alloy, a first-order, declarative language. We extended the analyzer to log (some of) its interactions with the user. We studied the interaction logs collected from several students and identified some opportunities for improving the analyzer, both the performance of analyses and the user interaction. We hope that this paper motivates the Alloy community to improve and use our logging extension and to further study the new interactions logs. We expect that such studies would lead to improving the analyzer for common usage scenarios, benefiting the entire community. More generally, we argue for a broader effort on studying the usage of software-development tools.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors*; D.2.6 [**Software Engineering**]: Programming Environments; D.2.2 [**Software Engineering**]: Design Tools and Techniques; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Human Factors, Design, Performance

Keywords

Alloy language, Alloy Analyzer, user interaction, incremental analysis, continuous analysis

*This paper is based on the work [5] presented at the Sixth Workshop on Language Descriptions, Tools, and Applications (LDTA 2006) in Vienna, Austria.

1. WHAT IN THE ALLOY ANALYZER?

The Alloy tool-set [2, 3] has been successfully used in research and teaching for several years and has assisted in finding and correcting flaws in software systems (e.g., [4, 7]). However, prior to our initial work [5], there had been no study into how users interact with the tool-set. We next present the motivation for doing our study, the logging that enabled it, and some results that it showed.

1.1 Need for the study

Two aspects of Alloy make studying user interactions with the tool-set particularly worthwhile: the declarative nature of the language and the bounded-exhaustive checking performed by the analyzer. Declarative logic paradigms, in general, and Alloy, in particular, tend to elicit a pervasive use of conjunction. An Alloy model is often built by first defining sets and relations that represent the model and then defining formulas that constrain the representation appropriately, starting from a minimal representation and incrementally strengthening it until a sufficient level of detail is attained. The use of the analyzer in an interactive fashion assists the users in making the incremental changes and checking their validity. These incremental changes tend to be small, so the analyzer may exploit the differences introduced between consecutive analyses to provide a faster analysis using the result of the previous analysis.

The Alloy Analyzer performs bounded exhaustive analysis: the analyzer checks a given formula for a specified *scope*, i.e., bound on the universe of discourse. The analyzer translates the Alloy model into propositional formulas and uses off-the-shelf SAT solvers to check the resulting formulas. The nature of the analyzer's checking implies that its results are valid with respect to the given scope only, i.e., if the analyzer fails to find an instance that satisfies an Alloy model within a given scope (bound), an instance may still exist in a larger scope. For Alloy users, it is natural to increase their level of confidence in a model by increasing the scope and re-checking the model after the analyzer failed to generate a desired instance in a smaller scope. Notice that in such a scenario, the only change in the model between two consecutive analyses is the scope—once again, a situation arises where the analyzer may be able to provide a faster checking using the result of the previous analysis.

Based on the above scenarios and our personal experience with the analyzer, we hypothesize that *incremental analysis*, which reuses results of previous analyses, could improve the performance of the Alloy Analyzer. To evaluate our hypothesis, we need to collect the actual logs of interaction with the analyzer.

1.2 Logging

To investigate how users work with the Alloy Analyzer, we have modified the analyzer to log (some of) its interactions with the user. We designed our logging to provide the Alloy tool-set developers with the usage data that could help improve the tool-set. The logging currently captures several build, analysis, and user-interface events. The build events record the compilation of the models (from Alloy to propositional formulas). The analysis events record the Alloy analysis commands used after a successful compilation. The logging stores all the information required to replay an event, including the configuration of the analyzer and the SAT solver, the Alloy file being compiled and any files referenced, and the representation of the analysis commands. The logging also stores the time stamps for event beginnings and ends. Finally, the logging stores other usage data that might further improve understanding of the usage pattern of the analyzer. For example, the user-interface events may help streamline the workflow of the analyzer, while the information about failed builds may shed light on the common mistakes that users make while learning the Alloy language and the ways in which the analyzer can help them develop correct models.

1.3 Results

We asked students in two of our graduate classes to solve a problem set that required them to build Alloy models using the analyzer extended with our logging. Our study [5] of the resulting 68 logs from 11 students shows three key observations.

Observation 1: The answer to "What will the user analyze next?" is often "Something similar to what the user has just analyzed." Users often perform consecutive analyses with slightly different models, as expected from the two afore-mentioned aspects of Alloy. This suggests that incremental constraint-solving techniques can improve the analyzer's performance. Alloy's use of SAT technology and recent advances in incremental SAT solving [8] provide a natural start for exploring optimizations for the Alloy Analyzer. We explored the use of incremental SAT for the problem set examples in the simple scenarios when the user adds a new fact to the model and when the user increases the scope by one. For these scenarios, the time to generate a solution reduces by up to a half [5].

Observation 2: It is sometimes possible to predict the answer to "What will the user do next in the analyzer?" User's interaction with the analyzer is sometimes predictable, e.g., that the user will compile and analyze the model or that the user will ask for additional solutions to the model. This points out that, similar to continuous compilation and continuous testing [6], the analyzer can continuously precompute the result of a future action while the user is editing the model or visually inspecting a solution. For the problem set examples, the time a user takes to inspect a particular solution is always less than the time the SAT solver takes to generate the next solution [5]. Thus, precomputing the next solution while the user is inspecting the current one can instantaneously generate a result when the users asks for the next solution.

Observation 3: The answer to "What will a beginner user do?" may be more surprising than the answer to "What will an expert user do?" (Beginner) users can naturally develop semantically equivalent (but syntactically different) Alloy models that have significantly different analysis time. Thus, it would be worthwhile to study manual and automatic transformations of Alloy models that could result in improved analysis time. This result also provides evidence against a previous claim [7] that semantically equivalent Alloy models tend to have similar analysis time. While it is clear that in any reasoning system the analysis time depends on the specific formulation of the problem, our result shows that beginner Alloy users naturally create models that have different analysis time [5].

1.4 Summary

Our results provide an encouraging starting point for the further studies of the Alloy Analyzer. We hope to engage the Alloy community to collect more logs and analyze them to detect potential further improvements for the analyzer. Our logging is included in the current, publicly available version of the analyzer. Our prototype incremental analysis, however, handles only very basic scenarios. Implementing full incremental analysis and continuous analysis in the analyzer would be important future steps in realizing the potential improvements.

2. WHAT IN OTHER TOOLS?

The Alloy Analyzer is only one example tool used in software development. We next propose a specific study for a completely different tool and then argue for more such studies in general.

Eclipse (`http://eclipse.org`) is an open-source IDE for Java and other languages. It has been downloaded tens of millions of times and used in many more interactive sessions. Eclipse does offer a logging facility, but it is primarily used for logging warning and error messages rather than user interactions. We specifically propose to investigate how developers in Eclipse use refactorings [1], which are program transformations that change the internal structure of the code without changing its external behavior, e.g., renaming a method. Eclipse provides an engine that automates refactorings. Developers using Eclipse mix manual code editing and automated refactorings. Two key questions are: "How (frequently) do developers use refactorings?" and "Can we predict when a developer will apply a refactoring?". Answering these questions could enable specializing refactorings for different users.

In conclusion, we consider studies of tool usages important in general. We have summarized our study of logs from the Alloy Analyzer and proposed a study of logs from Eclipse. We believe that collecting and analyzing detailed logs of user interactions could lead to improvement of these and many other tools.

Acknowledgments. We would like to thank the 11 students from our classes for submitting the logs of their interactions with the Alloy Analyzer. We would also like to thank Xiaoming Li, Daryl Shannon, and Jabari Walker for their help with our initial study and Derek Rayside and Greg Dennis for their help with Alloy logging.

3. REFERENCES

[1] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: Improving the Design of Existing Code*. Adison-Wesley, 1999.

[2] D. Jackson. *Software Abstractions: Logic, Language and Analysis*. The MIT Press, Cambridge, MA, 2006.

[3] D. Jackson, I. Schechter, and I. Shlyakhter. ALCOA: The Alloy constraint analyzer. In *Proc. ICSE*, Limerick, Ireland, June 2000.

[4] S. Khurshid and D. Jackson. Exploring the design of an intentional naming scheme with an automatic constraint analyzer. In *Proc. ASE*, Grenoble, France, Sep 2000.

[5] X. Li, D. Shannon, J. Walker, S. Khurshid, and D. Marinov. Analyzing the uses of a software modeling tool. In *Proc. LDTA*, Vienna, Austria, Apr. 2006. `http://www.ece.utexas.edu/~khurshid/papers/ldta06.pdf`.

[6] D. Saff and M. D. Ernst. An experimental evaluation of continuous testing during development. In *Proc. ISSTA*, Boston, MA, 2004.

[7] K. Sullivan, J. Yang, D. Coppit, S. Khurshid, and D. Jackson. Software assurance by bounded exhaustive testing. In *Proc. ISSTA*, Boston, MA, 2004.

[8] J. Whittemore, J. Kim, and K. Sakallah. SATIRE: A new incremental satisfiability engine. In *Proc. DAC*, Las Vegas, NV, June 2001.