Usability Barriers for Liquid Types (Summary of Published Work)

CATARINA GAMBOA, Carnegie Mellon University, USA and LASIGE, University of Lisbon, Portugal
ABIGAIL REESE, Carnegie Mellon University, USA
ALCIDES FONSECA, LASIGE, University of Lisbon, Portugal
JONATHAN ALDRICH, Carnegie Mellon University, USA

Extended Abstract

Liquid types [4] extend traditional type systems with logical predicates that allow developers to express complex properties in different applications. For example, they have been used to track data between different layers in MVC applications in Haskell [2], model typestate protocols in Java [1], and track the semantics of database queries in Rust [3]. Despite their expressive power and implementation in different languages, the general developer community has not yet adopted liquid types, which raises the question of the usability barriers to adopting and using liquid types.

In this paper, we present a study with 19 developers with different levels of expertise in using LiquidHaskell [5], the most mature implementation of liquid types now. Twelve developers were new to liquid types but familiar with the target language, while seven were experienced users who had used LiquidHaskell across different application areas. We used different qualitative research methods with these developers, including interviews, observations, retrospectives, and think-aloud protocols, depending on their expertise and the projects they could show us. This study identified nine barriers to adopting liquid types, spanning three themes: developer experience, scalability challenges with complex and large codebases, and understanding the verification process.

Verification barriers come from the unclear divide between the verification layer and the programming language, confusing verification features in liquid types, and the lack of familiarity with proof engineering. These challenges make it difficult for developers without a formal verification background to understand the verification process and how to use it effectively. The developer experience barriers compound the difficulties in understanding the verification process, since there is limited IDE support and learning resources, and the error messages can be unhelpful in diagnosing the problems. Setting up and installing the tools are also challenges that prevent developers from even starting to use liquid types. Finally, scalability challenges arise when working with large and complex codebases, where the verification often becomes slow, and internally, the SMT solver has limitations for certain types of properties. Additionally, the mix of automation and manual flexibility and the opaque use of the SMT solver make developers unsure of what is necessary to prove properties.

The barriers identified in this study can also be seen in other implementations of liquid types and even other verification techniques. Therefore, by addressing these usability barriers, we can enable more developers to adopt these techniques and create more robust and reliable software.

This paper was recently published at PLDI 2025 (https://dl.acm.org/doi/10.1145/3729327) and brings together topics on programming languages, software engineering, and human-computer interaction to create a comprehensive view of the usability challenges to the broader adoption of liquid types.

Authors' Contact Information: Catarina Gamboa, Carnegie Mellon University, Pittsburgh, USA and LASIGE, University of Lisbon, Lisbon, Portugal, cvgamboa@fc.ul.pt; Abigail Reese, Carnegie Mellon University, Pittsburgh, USA, aereese@andrew.cmu.edu; Alcides Fonseca, LASIGE, University of Lisbon, Lisbon, Portugal, amfonseca@fc.ul.pt; Jonathan Aldrich, Carnegie Mellon University, Pittsburgh, USA, jonathan.aldrich@cs.cmu.edu.

References

[1] Catarina Gamboa, Paulo Canelas, Christopher Steven Timperley, and Alcides Fonseca. 2023. Usability-Oriented Design of Liquid Types for Java. In *International Conference on Software Engineering (ICSE)*. IEEE, 1520–1532. https://doi.org/10.1109/ICSE48619.2023.00132

[2] Nico Lehmann, Rose Kunkel, Jordan Brown, Jean Yang, Niki Vazou, Nadia Polikarpova, Deian Stefan, and Ranjit Jhala. 2021. STORM: Refinement Types for Secure Web Applications. In *Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, 441–459. https://www.usenix.org/conference/osdi21/presentation/lehmann

[3] Nico Lehmann, Cole Kurashige, Nikhil Akiti, Niroop Krishnakumar, and Ranjit Jhala. 2025. Generic Refinement Types. *Proc. ACM Program. Lang.* 9, POPL (2025), 1446–1474. https://doi.org/10.1145/3704885

[4] Patrick Maxim Rondon, Ming Kawaguchi, and Ranjit Jhala. 2008. Liquid types. In *Programming Language Design and Implementation (PLDI)*. ACM, 159–169. https://doi.org/10.1145/1375581.1375602

[5] Niki Vazou, Eric L. Seidel, and Ranjit Jhala. 2014. LiquidHaskell: experience with refinement types in the real world. In *ACM SIGPLAN symposium on Haskell*. ACM, 39–51. https://doi.org/10.1145/2633357.2633366