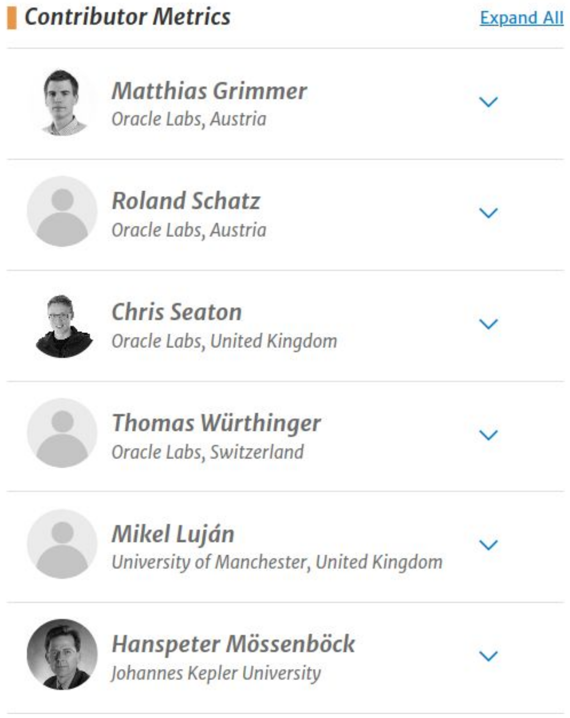
Cross-Language Interoperability in a Multi-Language Runtime

paper review

Meta data

- Authors: Matthias Grimmer, Roland Schatz, Chris Seaton, Thomas Würthinger, Mikel Luján, Hanspeter Mössenböck.
- Conference: ACM Transactions on Programming Languages and Systems , 2018
- 63 reference inside the paper.
- 15 citations.
- 1189 Downloads.
- 43 pages.


Contributor Metrics
Expand All

Matthias Grimmer
Oracle Labs, Austria

Roland Schatz
Oracle Labs, Austria

Chris Seaton
Oracle Labs, United Kingdom

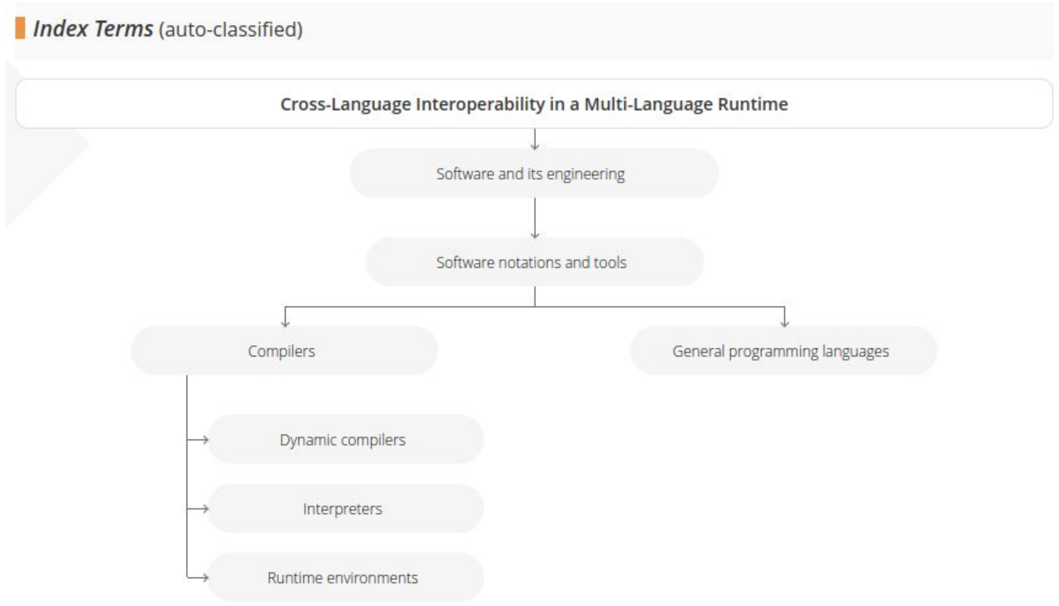
Thomas Würthinger
Oracle Labs, Switzerland

Mikel Luján
University of Manchester, United Kingdom

Hanspeter Mössenböck
Johannes Kepler University

Paper layout

1. Introduction
2. System overview
3. Generic Access
4. Case studies
5. Performance evaluation
6. Lesson learned
7. Related work
8. Conclusions
9. Acknowledgments


Index Terms (auto-classified)

Cross-Language Interoperability in a Multi-Language Runtime
Software and its engineering
Software notations and tools
Compilers
General programming languages
Dynamic compilers
Interpreters
Runtime environments

What did they do

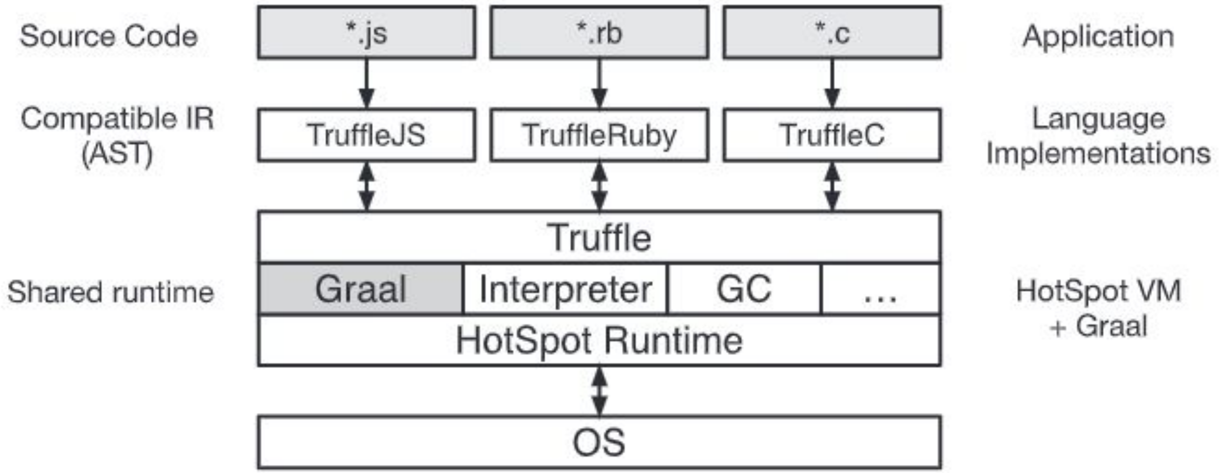
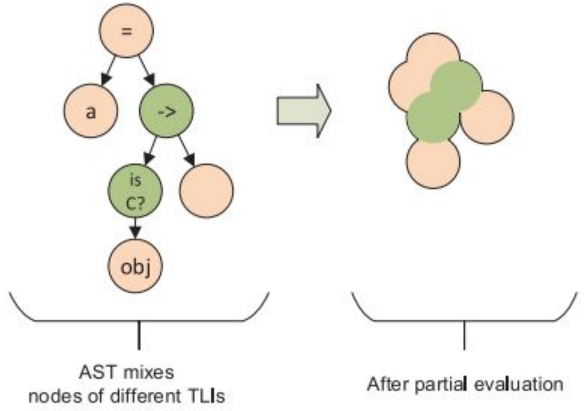


Fig. 3. Layers of the TruffleVM: TruffleJS, TruffleRuby, and TruffleC are hosted by the Truffle framework on top of the HotSpot VM (using Graal as a JIT compiler).

Features

- Pictures in the same style
- A lot of useful refs
- “Lesson learned”



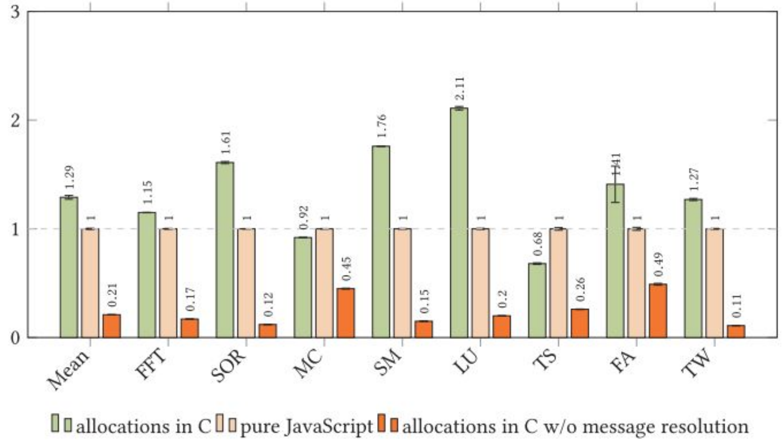


Fig. 16. Main part in JavaScript and allocations in C with and without message resolution (normalized to pure TruffleJS performance; higher is better).

Features

- Implementation effort
- Scale of the project
- Related work (p. 37)

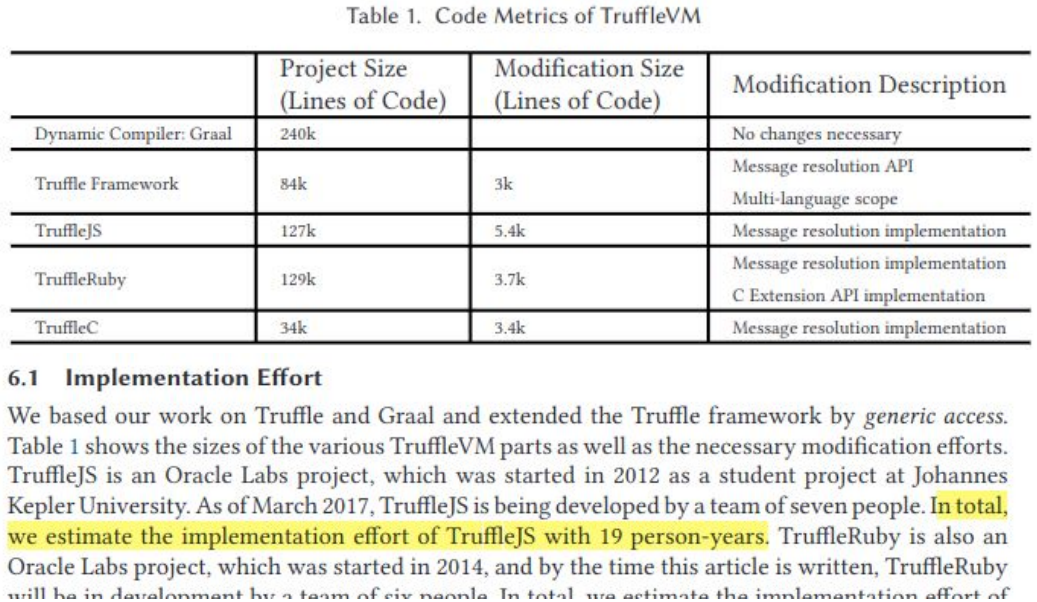


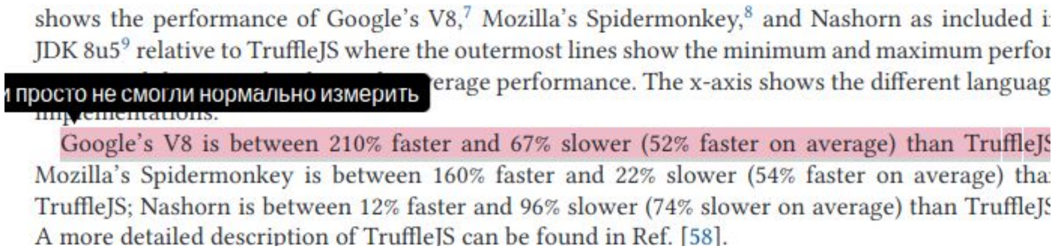
Table 1. Code Metrics of TruffleVM

	Project Size (Lines of Code)	Modification Size (Lines of Code)	Modification Description
Dynamic Compiler: Graal	240k		No changes necessary
Truffle Framework	84k	3k	Message resolution API Multi-language scope
TruffleJS	127k	5.4k	Message resolution implementation
TruffleRuby	129k	3.7k	Message resolution implementation C Extension API implementation
TruffleC	34k	3.4k	Message resolution implementation

6.1 Implementation Effort

We based our work on Truffle and Graal and extended the Truffle framework by *generic access*. Table 1 shows the sizes of the various TruffleVM parts as well as the necessary modification efforts. TruffleJS is an Oracle Labs project, which was started in 2012 as a student project at Johannes Kepler University. As of March 2017, TruffleJS is being developed by a team of seven people. In total, we estimate the implementation effort of TruffleJS with 19 person-years. TruffleRuby is also an Oracle Labs project, which was started in 2014, and by the time this article is written, TruffleRuby will be in development by a team of six people. In total, we estimate the implementation effort of

Defects


shows the performance of Google's V8,[7] Mozilla's Spidermonkey,[8] and Nashorn as included i
JDK 8u5[9] relative to TruffleJS where the outermost lines show the minimum and maximum perfor

и просто не смогли нормально измерить

erage performance. The x-axis shows the different languag
implementations.

Google's V8 is between 210% faster and 67% slower (52% faster on average) than TruffleJS
Mozilla's Spidermonkey is between 160% faster and 22% slower (54% faster on average) tha
TruffleJS; Nashorn is between 12% faster and 96% slower (74% slower on average) than TruffleJS
A more detailed description of TruffleJS can be found in Ref. [58].

My personal assessment by criteria

Problem statement.	8 / 10
Innovation.	8 / 10
Contribution.	8 / 10
Logical correctness.	8 / 10
Proof of statements.	8 / 10
Readability.	7 / 10

But if you just look at what they have done..