Rascal, 10 Years Later

Paul Klint
SWAT.engineering & CWI, Amsterdam
The Netherlands
paul.klint@swat.engineering

Tijs van der Storm
CWI, Amsterdam
University of Groningen, Groningen
The Netherlands
storm@cwi.nl

Jurgen Vinju
CWI & SWAT.engineering, Amsterdam
TU/e, Eindhoven
The Netherlands
jurgen.vinju@cwi.nl

***Abstract*—When we designed the first version of Rascal in 2009, we jokingly promised ourselves to only write a single paper on the language itself, and see it as vehicle for research from then on,—that one paper became the SCAM 2009 article [2], now awarded with the SCAM most influential paper award. Since then, Rascal has evolved significantly, and has been successfully applied in research, education, and industry. This extended abstract gives an overview of the impact of Rascal over the last 10 years, and looks at current and future developments.**

Language Evolution: Since 2009 Rascal has evolved significantly, with a more powerful grammar formalism, a refined module system for extensible programming, and a new function dispatch mechanism based on pattern matching. Furthermore, Rascal has been complemented by large sets of libraries for various programming tasks, such as visualization, web-based GUIs, statistics, EMF-based modeling, and others. Finally, Rascal includes a number of language front-ends (Java, PHP, Javascript, C/C++), enabling static source code analysis on industrial scale software projects.

Rascal has been called a "one-stop shop for meta programming"; as such Rascal is not merely a language anymore, but a complete environment, including integration with the Eclipse IDE. This allows DSL designers to use Rascal as a language workbench, obtaining IDE features for their DSL at very low cost.

Research: A primary goal of Rascal has been to be a vehicle for research in source code analysis and transformation, much like a telescope enables research in astronomy. As such it has been successful in enabling contributions in areas of software analysis, software evolution, language design, language engineering, model-driven engineering, parsing, domain-specific languages, static analysis, language workbenches, and others. Papers have been appeared in conferences like ICSE, PLDI, ICSME, ASE, ISSTA, MSR, ECOOP, OOPSLA, Onward!, ICPE, SCAM, <Programming>, SLE, and GPCE, including award-winning papers [1], [3], [4].

In general, this work can be divided in two categories: papers that used Rascal to perform research (e.g., as language prototyping tool or source code analysis framework), or papers that addressed specific aspects of Rascal itself, such as the internal data structures of the implementation, or new parsing algorithms. This reflects how Rascal plays a pivotal role in the Software Analysis and Transformation (SWAT) group at CWI.

Use in Teaching: Next to having enabled significant research results in the past 10 years, Rascal has been successfully used in teaching as well. Courses on software evolution at the University of Amsterdam, Open University, Eindhoven University of Technology, and University of Twente, have employed Rascal to allow students create their own tools for refactoring, code smell detection, clone detection, etc. Similarly, at University of Groningen, Rascal is used in a course on Software Language Engineering, where students learn about all aspects of engineering DSLs. Finally, Rascal has been used in tutorials on software analysis, language extension, DSL engineering for both academic and broader audiences.

Industry Impact: Rascal has been used in research collaborations with organizations like Philips Healthcare, Océ, ING Bank, and the Netherlands Forensics Institute (NFI). In 2017, our experience with Rascal on realistic projects, motivated the start of the CWI spin-off company SWAT.engineering[1], which puts Rascal to use for commercial language and software development projects.

Outlook: The CWI SWAT group is strongly committed to sustaining Rascal, and we expect it to continue to evolve and support research and education in the long term. Ongoing projects are the development of a new compiler, the integration of data-dependent parsing technology, code formatting based on machine learning, constraint-based type checking, supporting Microsoft's Language Server Protocol (LSP), and maturing the Rascal eco-system through a package manager and deployment system. To enable and invite contributions from different institutions and companies the Rascal open source project is facilitated by an independent organization called UseTheSource[2]. For more information, we refer the reader to `http://rascal-mpl.org`.

References

[1] F. Coulon, T. Degueule, T. van der Storm, and B. Combemale. Shape-diverse DSLs: Languages Without Borders (Vision Paper). In *SLE'18*, pages 215–219, 2018.

[2] P. Klint, T. van der Storm, and J. J. Vinju. Rascal: A domain specific language for source code analysis and manipulation. In *SCAM'19*, pages 168–177. IEEE, 2009.

[3] D. Landman, A. Serebrenik, and J. J. Vinju. Challenges for static analysis of Java reflection – literature review and empirical study. In *ICSE'17*. IEEE, May 2017.

[4] M. Steindorfer and J. J. Vinju. Performance modeling of maximal sharing. In *ICPE'16*, 2016.

[1] `http://www.swat.engineering`
[2] `http://usethesource.io`