Rust Toolchain for Event-B with Language Server Support

Denis Efremov[1] and Ilya Shchepetkov[2]

[1] Ivannikov Institute for System Programming of the RAS, Moscow, Russia
[2] Kaspersky Lab, Moscow, Russia
efremov@ispras.ru, ilya.shchepetkov@17451k.space

Abstract. We present a standalone toolchain providing Language Server Protocol support for Event-B. Written in Rust, it comprises a PEG-based parser with error recovery, an LSP server enabling Event-B development in VS Code, Neovim, and other editors, a bidirectional Rodin XML converter, and CLI tools for CI/CD integration. We have evaluated the parser on 68 publicly available Rodin projects, including industrial-scale specifications such as ARINC 653, ERTMS/ETCS, and HIMACF, the ABZ case studies, and cross-validated the results against Rodin's own parser.

Keywords: Event-B · Language Server Protocol · Rust · Formal Methods

1 Introduction

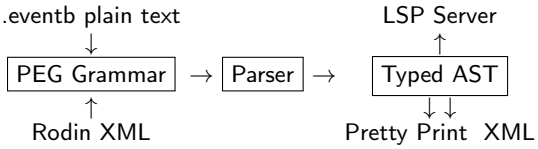
Event-B [1] is a formal method for system-level modelling widely used in safety-critical domains. The Rodin platform [2] provides the reference toolset, but its Eclipse architecture ties practitioners to a single IDE. CamilleX [4] introduced text-based editing within Rodin, but it still requires Eclipse/Xtext. Additionally, Eclipse-based solutions must be constantly maintained and updated to newer versions of the SDK, which is a costly and time-consuming process.

Modern industrial development relies on editors such as VS Code and Neovim, version control with text diffs, and CI pipelines, which are difficult to integrate with Rodin. We aim to bridge this gap with an open-source Rust toolchain [3] comprising a parser, an LSP server, an XML converter, various CLI tools, and integration with the ProB model checker.

2 Architecture

We parse Event-B models into a strongly-typed abstract syntax tree (AST) with source spans using a PEG grammar and the `pest` [9] parser. Rodin XML files (`.bum`/`.buc` and `.zip` archives) are also parsed into the same AST, enabling bidirectional conversion.

[3] The code will be published under MIT license in May, before the Rodin Workshop 2026.



The grammar covers the full Event-B language with Unicode and ASCII operator alternatives. Error recovery enables partial parsing of malformed input for responsive IDE feedback. The pretty printer roundtrips the AST back to the Event-B text and XML formats.

The `eventb-lsp` module implements the Language Server Protocol, enabling Event-B support in any LSP-capable editor:

- *editing* — completion (keywords, operators, identifiers, snippets), signature help, formatting;
- *navigation* — go-to-definition, find-references, workspace symbols, resolving across `EXTENDS`/`REFINES`/`SEES` chains;
- *refactoring* — renaming, ASCII↔Unicode conversion;
- *display* — semantic highlighting, hover, folding.

Although Xtext provides LSP support, CamilleX does not currently expose a standalone language server, remaining tied to Eclipse; our LSP server works with any compatible editor.

The `eventb-validate` module performs batch syntax validation with text or JSON output and non-zero exit codes, which is suitable for CI pipelines. `eventb-format` converts bidirectionally between Rodin `.zip` archives and `.eventb` plain text files. These enable a workflow: edit Event-B as plain text with IDE support, track changes in git, validate in CI, and convert to Rodin XML format to work with proofs. We evaluated the toolchain on 68 publicly available Rodin projects—including ARINC 653 [11], ERTMS/ETCS [7], HIMACF [3], ABZ case studies, and academic models—with successful parsing, roundtrip pretty-printing, and cross-validation against Rodin's built-in parser.

3 Conclusion

Industrial adoption of formal methods depends heavily on integration with existing development workflows. By providing LSP support, plain-text editing, git-friendly workflows, and CI integration, our toolchain makes Event-B accessible to practitioners who already work with VS Code, Neovim, and modern DevOps pipelines. We believe that meeting engineers in their familiar environments is key to broadening the Event-B user base, particularly in industry.

A further direction is LLM-based agent integration: active work on proof automation exists for Isabelle [6], Lean [10,5], and Coq [8], while Event-B has yet to benefit from similar advances. The programmatic AST access, LSP support and CLI tools offered by our toolchain lay the groundwork for closing this gap.

References

1. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press (2010). https://doi.org/10.1017/CBO9781139195881
2. Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: An open toolset for modelling and reasoning in Event-B. International Journal on Software Tools for Technology Transfer **12**(6), 447–466 (2010). https://doi.org/10.1007/s10009-010-0145-y
3. Devyanin, P.N., Kulyamin, V.V., Petrenko, A.K., Khoroshilov, A.V., Shchepetkov, I.V.: Integrating RBAC, MIC, and MLS in verified hierarchical security model for operating system. Proceedings of the Institute for System Programming of the RAS **32**(1), 7–26 (2020). https://doi.org/10.15514/ISPRAS-2020-32(1)-1
4. Hoang, T.S., Snook, C., Dghaym, D., Fathabadi, A.S., Butler, M.: The CamilleX framework for the Rodin platform. In: Rigorous State-Based Methods (ABZ 2021). Lecture Notes in Computer Science, vol. 12709, pp. 124–129. Springer (2021). https://doi.org/10.1007/978-3-030-77543-8_11
5. Hubert, T., Mehta, R., Sartran, L., Horváth, M.Z., Žužić, G., Wieser, E., Huang, A., Schrittwieser, J., Schroecker, Y., Masoom, H., Bertolli, O., Zahavy, T., Mandhane, A., Yung, J., Beloshapka, I., Ibarz, B., Veeriah, V., Yu, L., Nash, O., Lezeau, P., Mercuri, S., Sönne, C., Mehta, B., Davies, A., Zheng, D., Pedregosa, F., Li, Y., von Glehn, I., Rowland, M., Albanie, S., Velingker, A., Schmitt, S., Lockhart, E., Hughes, E., Michalewski, H., Sonnerat, N., Hassabis, D., Kohli, P., Silver, D.: Olympiad-level formal mathematical reasoning with reinforcement learning. Nature (2025). https://doi.org/10.1038/s41586-025-09833-y
6. Jiang, A.Q., Welleck, S., Zhou, J.P., Lacroix, T., Liu, J., Li, W., Jamnik, M., Lample, G., Wu, Y.: Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In: The Eleventh International Conference on Learning Representations (ICLR) (2023), https://openreview.net/forum?id=SMa9EAovKMC
7. Mammar, A., Frappier, M., Fotso, S.J.T., Laleau, R.: An Event-B model of the hybrid ERTMS/ETCS level 3 standard. In: Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ 2018). Lecture Notes in Computer Science, vol. 10817, pp. 353–366. Springer (2018). https://doi.org/10.1007/978-3-319-91271-4_24
8. Thakur, A., Tsoukalas, G., Wen, Y., Xin, J., Chaudhuri, S.: An in-context learning agent for formal theorem-proving. In: First Conference on Language Modeling (COLM) (2024), https://openreview.net/forum?id=V7HRrxXUhN, arXiv:2310.04353
9. Tiselice, D., Tauber, T.: pest: The elegant parser, version 2.7. https://github.com/pest-parser/pest (2023), accessed: 2026-03-15
10. Xin, H., Ren, Z., Song, J., Shao, Z., Zhao, W., Wang, H., Liu, B., Zhang, L., Lu, X., Du, Q., Gao, W., Zhang, H., Zhu, Q., Yang, D., Gou, Z., Wu, Z., Luo, F., Ruan, C.: DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search. In: The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24–28, 2025. OpenReview.net (2025), arXiv:2408.08152
11. Zhao, Y., Yang, Z., Sanán, D., Liu, Y.: Event-based formalization of safety-critical operating system standards: An experience report on ARINC 653 using Event-B. In: 26th IEEE International Symposium on Software Reliability Engineering (ISSRE). pp. 281–292 (2015). https://doi.org/10.1109/ISSRE.2015.7381821