A roadmap for the Rodin toolset*
Version 1.0: 12 June 2008

Jean-Raymond Abrial[1], Michael Butler[2], Stefan Hallerstede[2], and Laurent Voisin[3]

[1] ETH Zurich, Switzerland, jabrial@inf.ethz.ch
[2] University of Southampton, United Kingdom, {mjb,sth}@ecs.soton.ac.uk
[3] Systerel, France, laurent.voisin@systerel.fr

1 Event-B and the Rodin Platform

Event-B is a formal method for system-level modelling and analysis [1]. Key features of Event-B are the use of set theory as a modelling notation, the use of refinement to represent systems at different abstraction levels and the use of mathematical proof to verify consistency between refinement levels.

The Rodin Platform[4] [2] is an Eclipse-based [3] toolset for Event-B that provides effective support for refinement and mathematical proof. Keep aspects of the are

- support for abstract modelling in Event-B
- support for refinement proof
- extensibility
- open source

To support modelliing and refinement proofs Rodin contains a modelling database surrounded by various plug-ins: a static checker, a proof obligation generator, automated and interactive provers. The extensibility of the platform has allowed for the integration of various plug-ins such as a model-checker (ProB), animators, a UML-B transformer and a LaTeX generator. The database approach provides great flexibility, allowing the tool to be extended and adapted easily. It also facilitates incremental development and analysis of models. The platform is open source, contributes to the Eclipse framework and uses the Eclipse extension mechanisms to enable the integration of plug-ins.

2 Roadmap

In its present form, Rodin provides a powerful and effective toolset for Event-B development and it has been validated by means of numerous medium-sized case studies. Naturally further improvements and extensions are required in order to improve the productivity of users further and in order to scale the application of the toolset to large industrial-scale developments. We outline the main extensions to Rodin that we have planned for a four year time frame. The outline descriptions of these planned extensions are grouped into sections 2.1 to 2.5 as follows.

* The continued development of the Rodin toolset is funded by the EU research project ICT 214158 DEPLOY (Industrial deployment of system engineering methods providing high dependability and productivity) www.deploy-project.eu. The toolset was originally developed as part of the project IST 511599 RODIN (Rigorous Open Development Environment for Complex Systems).

[4] Available from www.event-b.org

2.1 Model construction

Rodin provides a structured editor for constructing and modifying models stored in the database. As mentioned above, this facilitates easy extension as well as incremental development and analysis of models. Rodin needs further improvement to make it easier to perform standard editing tasks such as text search, copy/paste and undo/redo. Rodin will be extended to provide refactoring facilities, such as identifier renaming, that can be applied not just to models but to proof obligations, proofs and other forms of elements. Better support for browsing refinement links between models will be provided, for example, allowing the refinements and abstractions of events to be followed down or up a refinement chain.

2.2 Scaling

Event extension: In many Event-B developments it is common to perform superposition refinement where existing model features are maintained and additional features are added (e.g., additional variables, invariants, events and additional guards and actions for existing events). Currently events can be inherited as a whole but not extended. Rodin will support event extension (or superposition) where only the additional features are defined in a refined event and the existing features are inherited.

Composition and decomposition: Composition and decomposition of models is essential for scalability. There are plans to support two styles of composition for Event-B in Rodin:

Style A Sub-models interact via shared variables
Style B Sub-models interact via synchronisation over events

Rodin will be extended to provide support for composing models as well as decomposing models according to these styles. The proof obligation generator will be extended to enable independent refinement of sub-models.

Team-based development: Support for composition and decomposition will go some way towards enabling team-based development. But there will still be situations where a team needs to access a common set of models. Rodin will be extended to support concurrent modification of developments by providing viewing of change conflicts and automated merge of changes. It will provide support for version control. Support to analyse the impact of multiple user modifications on proof will be investigated.

2.3 Extending the proof obligations and theory

Proof obligations: Event-B models will be extended to include external variables. The proof obligation for such variables is that they must be preserved via a functional gluing invariant between abstract and concrete external variables. Other forms of proof obligations will also be added to support different paradigms (concurrent, distributed, sequential systems). These include proof obligations for preservation of event enabledness and richer variant structures(such as pointwise ordering and lexicographic ordering) for convergence proof obligations.

Mathematical extensions: Rodin will be extended to support richer types such as record structures and user-defined data types including inductive data types. Appropriate automated and interactive proof support for richer types will be investigated and provded. Higher order provers should enable proof support for inductive datatypes. Users will be able to define operators of polymorphic type (but not use operator overloading) as well as parameterised predicate definitions. Support for disjointness constraints will be added.

2.4 Proof and model checking

Rodin provides an open architecture for proof in the form of a proof manager that can use a range of provers to discharge proofs and sub-proofs. The existing automated provers will be extended with more powerful decision procedures. The use of existing first order and higher order automated provers will be investigated. As mentioned already, higher order provers should enable proof support for inductive datatypes. The possibility of exploiting automated techniques such as SMT and SAT will be investigated. The facilities of the ProB model checker will be fully integrated into Rodin.

2.5 Animation

Prototype animation plug-ins already exist. The animation facilities will be extended to allow for greater automation of large animations to support regression testing of models. A clear API to the animation will be provided to allow for easy integration with graphical animation tools.

2.6 Process and productivity

Requirements Handling and Traceability: The interplay between informal requirements and formal modelling is crucial in system development and needs better tool support. Facilities for constructing structured requirements documents and for building links between informal and formal elements will be added to Rodin. These will support traceability between requirements and formal models. Support for recording validation of these links and for managing consistency under change to requirements and to formal models will be provided.

Document management: Currently, the B2Latex plug-in for Rodin generates a LaTeX version of an Event-B model. The structure of the document follows the structure of the model. For proper document generation tool support will be provided whereby users dictate the order in which parts of the model are presented. They should be able to write a document, structured according to their needs that includes parts of an Event-B project and that is automatically kept in synchrony with the models.

Automated model generation: Automatic generation of refinements will be investigated and appropriate tool support provided. More general modelling and refinement patterns, enabling greater reuse of modelling and refinement idioms, will be investigated and tool support provided. Code generation from models will be investigated. An indirect route for achieving code generation will be to generated classical B and use the existing code generators for classical B.

3 Tool development procedures

We are promoting a rigorous approach to the development of Rodin. Key features of Rodin, e.g., the static checker and the proof obligation generator, were specified formally before being implemented. Test procedures are developed in tandem with implementing. We are setting up rigorous specification and code review procedures. The development of new features should also follow this approach.

Many of the extensions listed above will first be implement as separate Eclipse plug-ins. When their general value and quality is assured, they will be incorporated into the platform release.

The management of platform release versions will be coordinated amongst the platform and plug-in developers. Facilities for importing existing developments into newer versions of Rodin will be provided. Support documentation and tutorial material for tool users and plug-in developers will continue to be improved and updated.

4 Concluding

Many of the Rodin extensions outlined above will be implemented as part of the DEPLOY project. However, we welcome support from other researchers and tool developers in elaborating and realising the roadmap. Furthermore, we anticipate that researchers will investigate and implement Rodin extensions that are not identified in our roadmap.

References

1. Jean-Raymond Abrial. Modelling in Event-B: System and software engineering. To be published by Cambridge University Press, 2008.
2. Jean-Raymond Abrial, Michael Butler, Stefan Hallerstede, and Laurent Voisin. An open extensible tool environment for Event-B. In Z. Liu and J. He, editors, *ICFEM 2006*, volume 4260, pages 588–605. Springer, 2006.
3. Erich Gamma and Kent Beck. *Contributing to Eclipse*. Addison Wesley, 2003.

Acknowledgements

We would like to thank all members of the RODIN and DEPLOY projects and others who are contributing to the toolset especially Jens Bendisposto, Dominique Cansell, Kriangsak Damchoom, Thai Son Hoang, Cliff Jones, Thierry Lecomte, Michael Leuschel, Farhad Mehta, Christophe Métayer, Colin Snook and Francois Terrier.