Contents

Partners

DEPLOY in Space

The DEPLOY project is investigating large scale deployment of formal methods in different industrial sectors. The goal is to manage the inherent complexity of today's systems by applying formal methods together with sector specific supporting technologies and methods. The role of supporting technologies and tools is crucial; decades of formal methods research and practice have shown that any system engineering methodology that fails to integrate with existing methods has almost no chance of success.

Space Systems Finland Ltd (SSF) is a Finnish SME which focuses on the development of mission and safety critical systems. SSF sets the ambitious goal of evaluating whether a fully tool supported formal model driven development methodology could be achieved within DEPLOY. This requires that all stages of a normal development process are supported: requirements engineering, architectural design, implementation, integration and validation. In addition, it would be desirable if support for dependability and safety analysis such as Failure Modes and Effects Analysis (FMEA) and traceability could be achieved.

At the outset of the project, SSF had several people with strong background in formal methods. Most engineers were, however, unfamiliar with Event-B and Rodin. The big challenge was to understand how formal methods could be integrated into the development process, which in turn required that we understand at least the following questions:

- How difficult is it to train engineers in formal methods?
- Where should formal methods NOT be applied?
- What is the level of tool and methodology support we can expect for different system development phases and can the expected productivity gains be realized?

During the last three and half year, SSF has been trying to answer these questions and more. One of our first priorities was to assess how quickly people with and without formal methods background could learn the basics of Event-B and the Rodin toolset. After performing several (but still a limited number) experiments within our company, we are convinced that people with a software engineering background can become independent medium level users of Event-B and Rodin with 1 - 2 person months of practice. It should be noted however that serious system development requires the involvement of at least one advanced user. There must be someone who understands the complexities and machinery of Event-B and Rodin, because all bigger developments will encounter problems which cannot be solved without this understanding.

A second priority of SSF has been to understand where formal methods can do the most good and where they should not be used. Obviously these questions cannot be exhaustively answered, but as SSF mostly develops embedded control systems the application domain is restricted. During the project, we have been able to model, using

Event-B and Rodin, complex control systems such as the on-board software of the MIXS/SIXS instruments of the BepiColombo satellite. Another significant achievement has been the modeling of a realistic Attitude and Orbital Control Systems, which in general are responsible for control over rotations about vertical, lateral and longitudinal axes of a spacecraft or aircraft. During the application of Event-B and the Rodin Platform, several challenges were identified:

- Scalability requires that formal methods support distributed and compositional development. Several engineers must be able to work on the same system concurrently. No realistic system is designed and developed by a single engineer. Distributed development is enabled by a workable notion of compositionality so that interfaces between components are clear.

- Structured data must be addressed in an efficient manner. Although our chosen domain is control intensive, abstraction mechanisms and modeling methods for structured data must be efficient. Instantiation with real data must be possible and the scalability of the approach cannot be limited by the data alone.

During the DEPLOY Project, several approaches have been developed to tackle these challenges including modularization plug-ins and new data type systems. Although these problems have not been solved completely to our satisfaction, there is a clear path forward for the remainder of the project. Formal methods clearly have a role in the development of mission and safety critical systems. Personally, I feel there are three main drivers:

- Programmable technology must be used to implement mission and safety critical systems, because the market demands flexible solutions.

- Regulatory pressure on validation of safety critical systems implemented with programmable technology is increasing the evidence burden on system developers.

- As in all other sectors, there is increasing cost pressure on mission and safety critical development. Adequate cost efficiency cannot be achieved in the long run without formal methods.

Addressing the challenges mentioned above will be the focus of SSF for the remaining time in DEPLOY. However, since we feel that formal methods are of long-term strategic importance, SSF will continue to investigate the use of formal methods within DEPLOY and beyond.

Timo Latvala
Space Systems Finland Ltd.

DEPLOY after DEPLOY

We in DEPLOY are making a special effort to ensure that the results of our project are available and widely used after the project ends on April 30, 2012.

The following project web sites will stay available after the project end:

- The project web site with all information about the project results - http://www.deploy-project.eu/

- The project publication repository containing a rich collection of models, tutorials, technical reports, scientific papers, teaching materials and experience

reports - http://deploy-eprints.ecs.soton.ac.uk/

- Also, the Event-B and Rodin platform web site http://www.event-b.org/ providing various sources of documentation for users and developers of the Rodin toolset, will be supported after the project. A DEPLOY handbook is being built at http://handbook.event-b.org/ - your feedback are welcome.

- The Rodin platform open source development site at http://sourceforge.net/projects/rodin-b-sharp/, providing various relevant mailing lists for discussions, will be supported after the project ends.

- The site collecting evidence to support the use of formal methods can be found at http://fm4industry.cetic.be. This site has been developed by DEPLOY and we plan to expand it further in the remaining time of the project.

The project members are actively developing all these sites now and will make all necessary efforts to make them usable and complete in the remaining time of DEPLOY.

We are working now on creating a not-for-profit body, registered as "Rodin Tools Ltd" which will serve as a focus for the support and further development of the platform and tutorial, training and workshop provision, after the end of DEPLOY.

Alexander Romanovsky
Project Coordinator

Formal Mind GmbH – a DEPLOY Spin-Off

One of the goals of the Deploy project was to encourage the commercialization of the research. We are happy to announce that in July 2011, the Düsseldorf-based Formal Mind GmbH (http://www.formalmind.com) was incorporated. The founding team consists of Michael Leuschel of University of Düsseldorf, his Ph.D. student Michael Jastram who initiated the spin-off, as well as Ph.D. students Daniel Plagge and Jens Bendisposto.

Formal Mind will initially offer services in the fields of validation and analysis, as well as requirements management and engineering. Central to the services offered are the Düsseldorf-based Open Source projects ProB and ProR, both which have been used during the DEPLOY project and are partially sponsored by it.

Formal Mind

Formal Mind is in charge of managing the DEPLOY documentation project (http://handbook.event-b.org), which has the aim of creating a Rodin Handbook. This effort is well underway. Customers of Formal Mind include Alstom and Clearsy, amongst others.

Update on the Rodin Platform

The current version of the platform is Rodin 2.2.2. The Rodin development team is now preparing Rodin 2.3 expected on Friday 16th September 2011.

Rodin Release 2.1.1 (released in March 2011)

See http://wiki.event-b.org/index.php/Rodin_Platform_2.1.1_Release_Notes.

The version 2.1.1 of Rodin is a maintenance release which includes mainly bug fixes. The improvements made on the Proving UI caused some slowdowns when searching hypotheses within windows versions of the platform. This was corrected in Rodin 2.1.1.

Rodin Release 2.2 (released in June 2011)

See http://wiki.event-b.org/index.php/Rodin_Platform_2.2_Release_Notes.

This version of the Rodin contains some improvements and new features, such as:

- New way to upgrade. With Rodin 2.2, a new way to upgrade the Rodin Platform to the latest version available appeared. Indeed, it is now possible to get an archive to update Rodin. This spells the end to long migrations and plug-in re-installations to newer Rodin releases.
- Improved general platform performance. The build time has been decreased by about 40% on average since Rodin 2.0.
  See http://wiki.event-b.org/index.php/Rodin_Performances.
- Proving. New tactics were implemented such as Negation Normal Form, Generalized Modus Ponens or even Case distinction from implication. A highlight command is now available from the Prover UI in all the proving views to highlight selections and easily search pattern occurrences in the manipulated predicates.

Rodin Release 2.2.2 (released in July 2011)

See http://wiki.event-b.org/index.php/Rodin_Platform_2.2.2_Release_Notes.

This version is a corrective release of the Rodin Platform which fixed two major bugs found in the early days of July.

Rodin Release 2.3 (expected in September 2011)

See http://wiki.event-b.org/index.php/Rodin_Platform_2.3_Release_Notes.

New features concerning the user-defined tactic profiles and auto-tactic tuning options are expected to be part of this release among maintenance corrections.

Update on the Rodin Plug-ins

Existing plug-ins continue to be maintained and developed.

Here we highlight a number of new plug-in developments.

New Rodin Structured Editor: The new Rodin Editor is an Event-B model editor, based on the same principles as the historical structured Event-B Editor. This latter editor shown its weakness while editing large models and beyond that didn't allow one to show some needed information such as the inherited elements that were formerly displayed only in the Pretty Print page. The aim of this new editor is to solve all these issues in a clean and easily readable manner, with enhanced keyboard navigation ability and easy edition.

See http://wiki.event-b.org/index.php/Rodin_Editor.

Event-B Statemachines plug-in: The Event-B Statemachines plug-in is part of the iUML-B tool that focuses on UML-B and Event-B integration. The plug-in provides a way of adding state machines directly to Event-B machines and is capable of translating former to Event-B language. It also offers a UML-like diagram editor for state machines, as well as state machine animation, which can be installed as additional plug-in that runs on top of ProB Animator.

See http://wiki.event-b.org/index.php/Event-B_Statemachines

Transformation patterns plug-in: The Transformation patterns plugin allows users to write transformation scripts in EOL (simple object-based imperative language) and easily run them over models within the same project. Editing is done in a simple text editor with syntax coloring. The plugin performs the necessary setup for running the transformations, and provides a simple library handling user input of Event-B elements such as machines, events, variables etc.

See http://wiki.eventb.org

Export to Isabelle: The "Export to Isabelle" plug-in allows users to export proof obligations and sequents to Isabelle/HOL. Its main purpose is to help researchers analyse proof obligations in Isabelle; in the long run, it will form the basis of automated proof tactics for Event-B based on Isabelle. The logic of Event-B's logic has been embedded in Isabelle.

See http://wiki.event-b.org/index.php/Export_to_Isabelle

B-Motion Studio: It is often very important to be able to show a formal model to a domain expert or manager, not versed in formal methods. For example, only a domain expert will be able to detect certain mistakes in the formal model. To easily and quickly build graphical visualisations of Rodin models, we have developed B-Motion Studio. The B-Motion Studio comes with a graphical editor to arrange graphical components and link them with the formal model. No new programming language has to be learned: the linking is described in B itself. To run a graphical visualisation, the ProB animator is used.

See http://cobra.cs.uni-duesseldorf.de/bmotionstudio/

The ProR Platform for Requirements Engineering and Formal Modeling: ProR is a tool for requirements engineering that supports the RIF 1.2 Standard natively. It is Eclipse-based and is built for extensibility. One of the primary goals of Deploy is the traceability between natural language requirements and formal models (initially Event-B Models). For this, we provide an integration plugin for the Rodin Platform to support traceability.

See: http://pror.org

The most up to date information on all plug-in developments can be found on the Event-B wiki. See http://wiki.event-b.org/index.php/Current_Developments

Michael Butler
University of Southampton

Focus on two tools: BMotion Studio and ProR

BMotion Studio

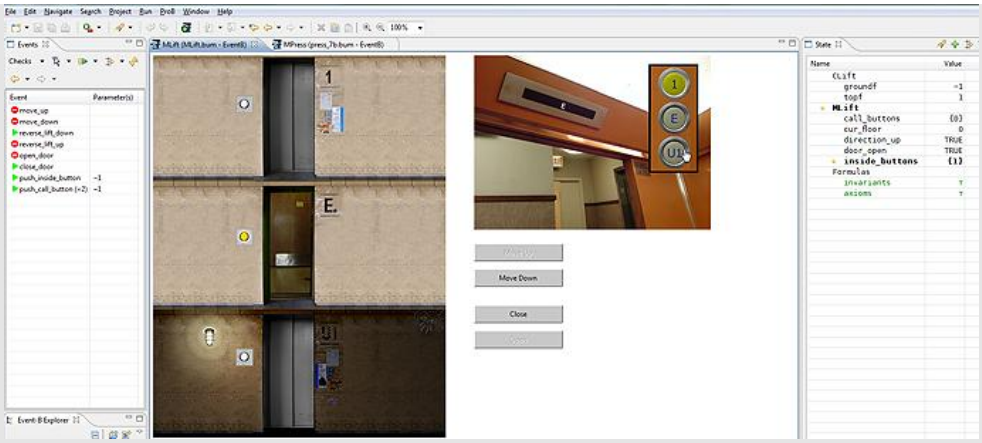
BMotion Studio (http://cobra.cs.uni-duesseldorf.de/bmotionstudio) is a graphical editor based on ProB (http://www.stups.uni-duesseldorf.de/ProB) which enables the developer of a Event-B model to set-up a domain specific visualization.

The main features of BMotion Studio are:

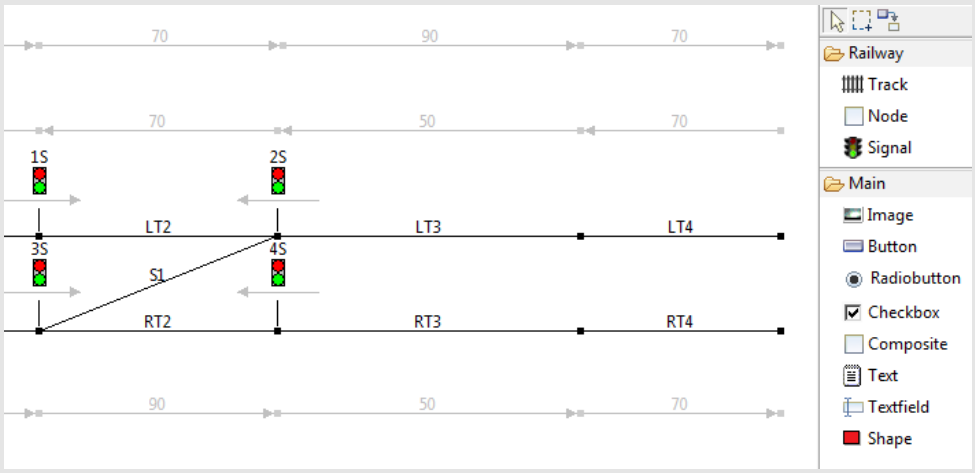
- The modeller stays within a single notation. BMotion Studio uses Event-B predicates and expressions as gluing code.
- An easy to use graphical editor, that allows to create visualisations within a few mouse clicks.
- A lot of possibilities for add-ons, to implement domain specific visual components.

BMotion Studio uses two concepts that are based on the model-view-control design pattern: Controls and Observers. A control is a graphical representation of some aspect of the model. For instance, if we model a traffic light with the two signals green and red, we might simply use two circle shapes for both signals. Observers define the Control's behaviour based on the information about the model's state, for instance what image it displays or which position it has. A series of experiments have shown that visualizations can be developed very quickly, often within a couple of hours.

Bellow a screenshot of BMotion Studio in action. The user can perform a state change by using ProB, i.e. by double clicking on an enabled event in the Events View or by using BMotion Studio, i.e. by clicking on a button which is wired to an event. As the state changes, the Observers start to evaluate expressions or predicates using ProB. The results are used to change the visualization.



As already mentioned before, BMotion Studio is designed for extensibility. We are currently developing such components for visualizing railway track layouts as shown in the screenshot below. If you are interested in contributing your own Addons/Plugins for BMotion Studio, check the Developer Documentation (not completed yet).



For more information about BMotion Studio check the corresponding webpage. Currently we are working hard on populating the website. However, you will find many useful information about BMotion Studio like a short tutorial (http://cobra.cs.uni-duesseldorf.de/bmotionstudio/index.php/Tutorial) and a few download-able examples (http://cobra.cs.uni-duesseldorf.de/bmotionstudio/index.php/Download) by now.

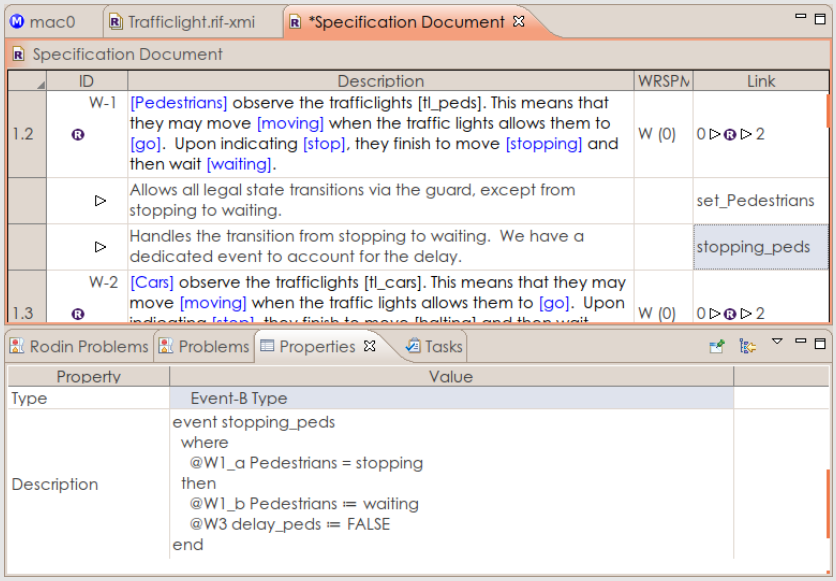
ProR

When we set out to improve requirements traceability within Rodin, we started with ambitious goals: rather than evolving the existing (now defunct) requirements plugin, we started from scratch, using the emerging Requirements Interchange Format (RIF/ReqIF) as a starting point. RIF was initially developed by the German automotive industry and is now an official OMG standard, renamed into ReqIF (http://www.omg.org/spec/ReqIF). ReqIF evolved far beyond data exchange, extending to new and existing tools, requirements servers, conversion tools, report generators and much more.

In RIF, requirements are called SpecObjects. These are typed, and the type determines the attributes of the SpecObject. A SpecObject can have an arbitrary number of attributes, like ID, description, status values, etc. SpecObjects are arranged hierarchically in Specifications. SpecRelations allow the creation of annotated links between SpecObjects.

ProR (http://pror.org/) is based directly on the RIF 1.2 data model (ReqIF 1.0 support is planned). Therefore, no conversion is necessary to read or write RIF, giving the tool immediate interoperability with tools like IBM Rational DOORS or MKS Integrity.

ProR is designed for extendibility, and we created a Traceability Plug-in (http://pror.org/content/tutorial-4-rodin-integration) that allows the integration with Rodin. It can highlight Event-B elements in the natural language requirements and allows linking to Event-B elements via drag and drop. The screenshot shows how the requirement W-1 is linked via an annotated SpecRelation to the event stopping_peds. The property pane shows the details of that event, as it is the target of the SpecRelation, which is selected in the main editor.



We managed to develop ProR relatively quickly, because we collaborated with another research project, ITEA Verde (http://www.itea-verde.org). Itemis, a Verde partner, developed a RIF core that was based on the Eclipse Modeling Framework (EMF). We started development with our own EMF-based core, which we later swapped out for the core from Itemis. The collaboration with ITEA Verde, and Itemis in particular, has been quite fruitful.

To ensure that ProR survives beyond the DEPLOY project, we submitted our work to the Eclipse Foundation, where it is currently in the review phase under the name Requirements Modeling Framework (RMF). We welcome your feedback and comments in the Eclipse Proposal forum (http://www.eclipse.org/forums/index.php/t/220570) and hope that it will soon be a full-grown Eclipse Project. We plan to offer commercial services around ProR via Formal Mind, a spin-off from the University of Düsseldorf.

Workshop on Refinement Based Development of Software Systems using Event-B Bangalore, 15-16 July 2011

Members of the DEPLOY Project made a major contribution to this workshop in Bangalore with the main speakers being Michael Butler and Colin Snook from the University of Southampton.

This workshop provided an intensive introduction to Event-B and the Rodin platform.

It covered modelling, refinement and proof in Event-B and UML-B. Other speakers were S. Ramesh and Manoranjan Satpathy from General Motors R&D Bangalore. Several representatives from Indian industry attended including GM and nuclear and space industries as well as several Indian Universities.

The workshop was organised jointly by IISC Bangalore (Deepak D'Souza) and GM R&D (P. Sampath, M. Sathpathy and S. Ramesh).

See: http://drona.csa.iisc.ernet.in/~deepakd/B-Workshop-2011/

DEPLOY Federated Event
Fontainebleau, 27 February-1 March 2012

A Federated event, gathering:

- A one-day tutorial for plug-ins developers
- A workshop for Rodin Users and Developers
- An industry day

will be organized next year at the end of the DEPLOY project. This event, hosted by IUT Sénart-Fontainebleau with the help of LACL laboratory, will take place in the south of Paris. It will be the occasion to draw a complete picture of the current status of the Rodin platform, the ongoing research and industrial use.

You are invited to submit contributions to the workshop and to the industry day. More details (program, venue, etc.) are available on the dedicated webpage.

See http://www.bmethod.com/php/federated-event-2012-en.php