Design and verification using Rodin

Michael Butler

University of Southampton

www.deploy-project.eu
www.event-b.org
Talk

• Example and Quick Demo

• Event-B and the Rodin tool

• Deploy Project

• Tool Challenges
Contributors to Rodin

Jean-Raymond Abrial  Laurent Voisin
Stefan Hallerstede  Thai Son Hoang
Farhad Mehta  Christophe Metayer
Thierry Lecomte  Michael Leuschel
Mathieu Clabaut  Colin Snook
Jens Bendisposto  Kriangsak Damchoom
Dominique Cansell  Cliff Jones
Renato Silva  Francois Terrier
Michael Jastram  Fabian Fritz
Issam Maamria  Andy Edmunds
Abdolbaghi Rezazadeh  Mar Yah Said
...

Rodin Design Philosophy

• Formal modelling is essential for *understanding* and *reasoning* when designing complex systems

• Design requires formal modelling at *multiple levels of abstraction* forming refinement chains

• Role of *proof* in design:
  – helps to improve understanding (invariants, variants)
  – helps to improve models (consistency, encourage simplicity)

• Construction and proof of refinement chains requires *strong incremental tool support*
Talk

- Example and Quick Demo
- Event-B and the Rodin tool
- Deploy Project
- Tool Challenges
Example: authorisation system

• Users are authorised to engage in activities
• User authorisation may be added or revoked
• Activities take place in rooms
• Users can only be in a room if they are authorised to engage in all activities that can take place in that room
Access control invariant:

\[\text{location}(u) = r \implies \text{takeplace}[r] \subseteq \text{authorised}[u]\]
Enter event

Enter(u,r) ≜
when
  grd1 : u ∈ User
  grd2 : r ∈ Room
  grd3 : takeplace[r] ⊆ authorised[u]
then
  act1 : location(u) := r
end

Does this operation maintain the security invariant?
Remove authorisation

RemoveAuth(u,a) \equiv
when
  grd1 : u \in User
  grd2 : a \in Activity
  grd3 : u \mapsto a \in authorised
then
  act1 : authorised := authorised \setminus \{ u \mapsto a \}
end

Does this operation maintain the security invariant?
Rodin demo

• illustrate interaction between modelling and proof
ProB model checking finds a counterexample

This shows a trace of operations leading to a violation of the invariant

This counterexample is found by the ProB tool

\[\text{auth} = \{\}, \text{act}(\text{Room1, Activity1}), \text{in}(\text{User1, Room1})\]
Refine model to introduce tokens

Abstract condition on a user and room for entering
\[ \text{takeplace}[r] \subseteq \text{authorised}[u] \]

is replaced by a condition on a token
\[ t \in \text{valid} \land \text{room}(t) = r \land \text{holder}(t) = u \]
Refine model to introduce tokens

To ensure consistency of the refinement we need invariant:

\[
\begin{align*}
t \in \text{valid} \\
\Rightarrow \\
\text{takeplace} [ \text{room}(t) ] \subseteq \text{authorised}[ \text{holder}(t) ]
\end{align*}
\]
Rational design – what, how, why

• **What** does it achieve?
  \[
  \text{location}(u) = r \\
  \implies \text{takeplace}[r] \subseteq \text{authorised}[u]
  \]

• **How** does it work?
  \[
  t \in \text{valid} \land r = \text{room}(t) \land u = \text{holder}(t)
  \]

• **Why** does it work?
  \[
  t \in \text{valid} \\
  \implies \\
  \text{takeplace}[\text{room}(t)] \subseteq \text{authorised}[\text{holder}(t)]
  \]
Talk

- Example and Quick Demo
- Event-B and the Rodin tool
- Deploy Project
- Tool Challenges
Event-B

- State-transition model (like ASM, B, VDM, Z)
  - set theory as mathematical language
- Refinement
  - state reification
  - one-to-many event refinement
  - new events (stuttering steps)
- Proof method
  - Refinement proof obligations (POs) generated from models
  - Automated and interactive provers for POs
  - Proof feeds back into modelling
Rodin Tool for Event-B
www.event-b.org

• Extension of Eclipse IDE (Java based)
• Open source – managed on sourceforge
  >2000 downloads of version 0.8.2
• Repository of *modelling elements*
  – Abstract syntax as Java objects, XML files
  – Allow for easy extension of abstract syntax
• Rodin Eclipse *Builder* coordinates:
  – Well-formedness + type checker
  – PO generator
  – Proof manager
  – Propagation of changes
Differential proving

When a model changes, proof impact of changes is minimised as much as possible:

- Syntactic comparison of POs

- Sufficiency comparison of POs
  - In case of success, provers return list of used hypotheses
  - Proof valid provided the used hypothesis are in the new version of a PO

- Variable renaming applied to models and corresponding POs and proofs (avoiding name clash)
Rodin Proof Manager (PM)

- PM constructs proof tree for each PO
- Automatic and interactive modes
- PM manages used hypotheses
- PM calls **reasoners** to
  - discharge goal, or
  - split goal into subgoals
- Collection of reasoners:
  - simplifier, rule-based, decision procedures, ...
- Basic tactic language to define PM and reasoners
Extensibility: Rodin Plug-ins

- Linking UML and Event-B
  - Colin Snook + Butler (Southampton)
- ProB: animation, consistency and refinement checking
  - Michael Leuschel + team (Düsseldorf)
- Graphical model animation
  - Brama (Clearsy)
  - AnimB (Christophe Metayer)
- Others under development ...
Talk

• Example and Quick Demo

• Event-B and the Rodin tool

• Deploy Project

• Tool Challenges
DEPLOY Integrated Project
2008 to 2012

- Industrial deployment of advanced system engineering methods for high productivity and dependability

- 7 Industrial Partners
  - Bosch, Siemens, SAP, SSF, Systerel, Clearsy, Cetic

- 5 Academic Partners
  - Newcastle, Aabo, Düsseldorf, ETHZ, Soton

www.deploy-project.eu
Industrial deployment partners

The industrial deployment will be in 4 major sectors

- **Bosch**: automotive
  - Cruise control
- **Siemens**: rail transportation
  - Computer-based train control
- **Space Systems Finland**: space systems
  - BepiColombo probe instrument software
- **SAP**: business information
  - Business service engineering
DEPLOY Goals

- **Understand** and **justify** the role of formal engineering methods in building dependable systems including advances in
  - requirements management and evolution
  - composition and reuse
  - role of formal methods in fault-tolerance, reliability, security

- **Address the barriers** to deploying formal engineering methods in industry and develop **deployment strategies**

- **Achieve** **deployment** of formal engineering methods

- **Scale** and **professionalise** Rodin technology
Talk

- Example and Quick Demo
- Event-B and the Rodin tool
- Deploy Project
- Tool Challenges
Tool Challenges

- Extending POs and mathematical language
- Proof and model checking
- Scaling
- Process and productivity
Extending POs and theory

- Range of paradigms (concurrent, distributed, sequential)
  - Enabledness: e.g., C1 enables A, C2 enables A
  - Convergence: more ordering relations, e.g., vectors

- Mathematical extension
  - Data types including inductive types
  - Polymorphic operators + theories
  - Polymorphic predicate definitions
Proof and model checking

- Prover improvements
  - More powerful decision procedures
  - Links to first order provers
  - Links to higher order provers (e.g., for polymorphic theories)
  - Links to SMT
  - Automated invariant + variant discovery

- More powerful model checking
  - Symmetry reduction, SAT

- Automated validation
  - Model animation
  - Regression testing of models

- Independent proof checking
Scaling

• Composition + decomposition
  – Support for composing models
  – Support for decomposing models and independent refinement

• Team-based development
  – Parallel development: viewing conflicts / automated merge
  – Version control
  – Impact on proof
  – Role of decomposition
Process and Productivity

- Requirements structuring and traceability
- Document management
- Modelling, refinement and proof patterns
  - Event-B and UML-B
  - Decision support
- Link with Classical B / AtelierB
- Code generation, assertion generation
  - C, ADA, SparkADA, Java, JML, Simulink, ...
Concluding

- Mastering design complexity through formal modelling and analysis
  - Encourage abstraction
  - Focus on what a system does
  - Focus on key / critical features
  - Incremental analysis and design

- DEPLOY + Rodin
  - Industrial deployment of methods and tools
  - Focus on early stage design
  - Tool is strong but much to be done