Formal and Semi-Formal Methods
Formal Verification through Model Transformation

James Williams

11th January 2010
Outline

Formal Methods

Semi-Formal Methods

Integrated Methods

Tools/Approaches

Conclusion
The Scenario

- Software has been constructed for decades
- Vast amounts of time and money spent fixing broken software
- Lots of research into best practices

- Verification - “Are we building the product right?” [Boe79] in [Som01]
Formal Methods

- Mathematically-based languages, techniques and tools
- Specification and verification
- The ability to prove
  - That a specification is realisable
  - Behavioural/structural properties
  - That a system has been implemented correctly
- Discover inconsistencies, ambiguities

[Win90, CW96]
Formal Methods

Industrial Concerns

- Entry cost to FMs high
- Insufficient tool support
- Lack of expertise/training
  - Specifications easy to understand, difficult to create [SB01]
- However...
  - (Possible) recent increase in the usage of FMs
  - Projects that have used FMs showed improvements in time, cost and quality
- Not every domain NEEDS formal methods
  - Mostly used in Transport, Finance and Defence
  - For Real-Time, Distributed, and Transaction processing applications

[CW96, WLBF09]
Semi-Formal Methods

- Model-Driven Engineering
  - Focus on creating *models* of a system at each stage in the development lifecycle
  - Automatic model transformations (e.g. to code)
- Intuitive, graphical notations
- Good at abstracting away detail
- Well-defined methodologies
- Produces more maintainable software

[Sch06, Wat08]
Semi-Formal Methods
Industrial Acceptance

- Widely used in industry
- Often informally defined semantics, causing:
  - Ambiguity
  - Inconsistency
  - Imprecision
  - Unable to be formally reasoned about

[KER99, HR04]
Semi-Formal Methods
Inconsistency Example
Semi-Formal Methods

Inconsistency Example

1. calculateTotal()
2. addOrder(o)
3. dispatch()
Semi-Formal Methods
Inconsistency Example
Inconsistency Example

1. calculateTotal()
2. addOrder(o)
3. dispatch()
Integrated Methods

- Take the best from both worlds
- Formal methods
  - Make precise, unambiguous statements
  - Formal reasoning
- Semi-formal methods
  - Speed, intuitive design
  - Manage size and complexity
- Approaches: side-by-side, formal links
- Levels: metamodel, semantic, methodical, tool

[SFD92, BLSS00]
Integrated Methods
Model Transformations for Verification

- Use model transformations to verify a model
  - I.e. transform into a format which can be reasoned about
  - Diagram → formal language
    - Usually need to add extra or adopt specific semantics

[TTSE08]
Tools/Approaches

- **UML → Z**
  - RoZ [DLCP00]
  - GeFoRME [Ama07]
  - AUtoZ [WP09]
  - Harmony (UML → Z++) [DH02]
  - UML → Object-Z [KBC05]

- **UML → B**
  - Ledang [LS02]
  - U2B [SB01]
  - Rodin UML-B [SB06]

- **UML → Alloy**
  - UML2Alloy [ABGR07]
Tools/Methods

UML → Z

- **Z**: Set theory and first order predicate calculus [Org02]
  - *Schema* structuring mechanism
  - Focuses on structural and functional properties
- **GeFoRME / AUtoZ** [Ama07, WP09]
  - Template based approach
  - Flexible semantics
  - Class and state diagrams
Tools/Methods
UML → B

- **B**: Abstract 'Machines' [Abr96]
  - Focus on refinement towards an implementation
- **U2B / Rodin UML-B** [SB01, SB06]
  - Graphical formal language
  - Event-B semantics
  - Class-like diagrams
Tools/Methods

UML → Alloy

▶ **Alloy**: Fully-automated analysis of models [Jac06]
  - Simulation and consistency checking
▶ **UML2Alloy** [ABGR07]
  - UML class diagram (inc. OCL constraints) → Alloy specification

```
PHDSTUDENT
name : NAME
pass()
fail()
sendThesis()
research()

ASSESSOR
scariness : Int
* 1

PHDSTUDENT0

PERSON
name : NAME

PHDSTUDENT1

NAME

name
name
name

ASSESSOR
scariness

17 / 26
```
## Tools/Methods

Comparison : Syntactic Completeness

<table>
<thead>
<tr>
<th>Method/Tool</th>
<th>Syntax Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoZ</td>
<td>Class</td>
</tr>
<tr>
<td>GeFoRME</td>
<td>Class, State, Object</td>
</tr>
<tr>
<td>AUtoZ</td>
<td>Class, State</td>
</tr>
<tr>
<td>Harmony</td>
<td>Class, State</td>
</tr>
<tr>
<td>UML → Object-Z</td>
<td>Class</td>
</tr>
<tr>
<td>Ledang</td>
<td>Class, Collaboration, State, OCL</td>
</tr>
<tr>
<td>U2B</td>
<td>Class</td>
</tr>
<tr>
<td>Rodin UML-B</td>
<td>Class, State</td>
</tr>
<tr>
<td>UML2Alloy</td>
<td>Class, OCL</td>
</tr>
<tr>
<td>Method/Tool</td>
<td>Semantic Flexibility</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>RoZ</td>
<td>Fixed / User-defined</td>
</tr>
<tr>
<td>GeFoRME</td>
<td>Flexible</td>
</tr>
<tr>
<td>AUtoZ</td>
<td>Flexible</td>
</tr>
<tr>
<td>Harmony</td>
<td>Fixed</td>
</tr>
<tr>
<td>UML → Object-Z</td>
<td>Fixed</td>
</tr>
<tr>
<td>Ledang</td>
<td>Fixed</td>
</tr>
<tr>
<td>U2B</td>
<td>Fixed</td>
</tr>
<tr>
<td>Rodin UML-B</td>
<td>Fixed</td>
</tr>
<tr>
<td>UML2Alloy</td>
<td>Fixed</td>
</tr>
</tbody>
</table>
None have complete source language (UML) coverage
Often need to annotate in the formal language
Few approaches consider the round-trip
  ▶ There is a focus on transforming the diagram into a formal specification
  ▶ What about updating the diagram with information discovered during the formal analysis? (Allows modellers to benefit)
  ▶ Use FMs “under the hood” [WLBF09]
Both formal and semi-formal methods have shortcomings

Integrating the two might improve the quality of software development

- Need to help modellers/designers determine the best FM for their task [KHR08]

Many approaches proposed

- None are complete (syntactically or semantically)
- None will be appropriate for all situations

We also need formal support for Domain Specific Languages
Thanks for listening

Slides and bibliography available at http://www-users.cs.york.ac.uk/~jw

Any questions?
## References I

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UML2Alloy: A challenging model transformation.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>J.R. Abrial.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The B-book: assigning programs to meanings.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N Amalio.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generative frameworks for rigorous model-driven development.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P. Braun, H. Lotzbeyer, B. Schatz, and O. Slotosch.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Consistent integration of formal methods.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B. W. Boehm.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Software engineering: R &amp; D trends and defense needs.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E.M. Clarke and J.M. Wing.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Formal methods: State of the art and future directions.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S. Dascalu and P. Hitchcock.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>An approach to integrating semi-formal and formal notations in software specification.</strong></td>
</tr>
</tbody>
</table>
References II

S. Dupuy, Y. Ledru, and M. Chabre-Peccoud.
An overview of roz: a tool for integrating uml and z specifications.

D. Harel and B. Rumpe.
Meaningful modeling: what’s the semantics of "semantics"?

D. Jackson.
*Software Abstractions: Logic, Language and Analysis*.

S. Kim, D. Burger, and D. Carrington.
An MDA approach towards integrating formal and informal modeling languages.

S. Kent, A. Evans, and B. Rumpe.
UML semantics FAQ.
*Object-Oriented Technology, ECOOP*, 1999.

F. Kordon, J. Hugues, and X. Renault.
From model driven engineering to verification driven engineering.
H Ledang and J Souqires.
Integration of UML and B specification techniques: Systematic transformation from OCL expressions into B.

International Standards Organisation.

C. Snook and M. Butler.

C. Snook and M. Butler.

D.C. Schmidt.

References IV

I Sommerville.
*Software Engineering.*

E. Turner, H. Treharne, S. Schneider, and N. Evans.
*Automatic generation of CSP—— B skeletons from xUML models*, pages 364–379.

A. Watson.
A brief history of MDA.
*Upgrade, the European Journal for the Informatics Professional*, 9(2), 2008.

J.M. Wing.
A specifier's introduction to formal methods.

J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald.
Formal methods: Practice and experience.

J.R Williams and F. Polack.
Automated formalisation for verification of diagrammatic models.