Formal and Semi-Formal Methods

Formal Verification through Model Transformation

James Williams

11th January 2010

Outline

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

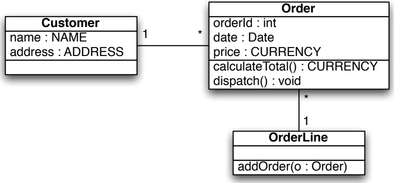
Industrial Acceptance

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

[KER99, HR04]

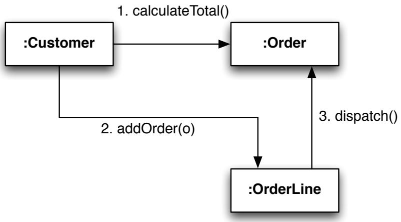
Semi-Formal Methods

Inconsistency Example



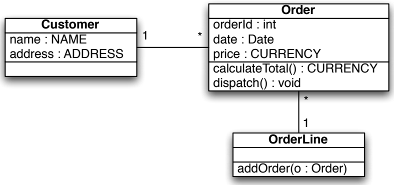
Semi-Formal Methods

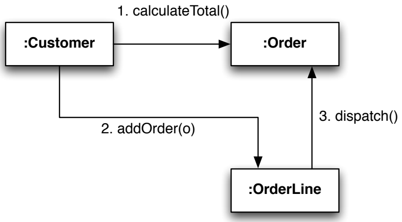
Inconsistency Example



Semi-Formal Methods

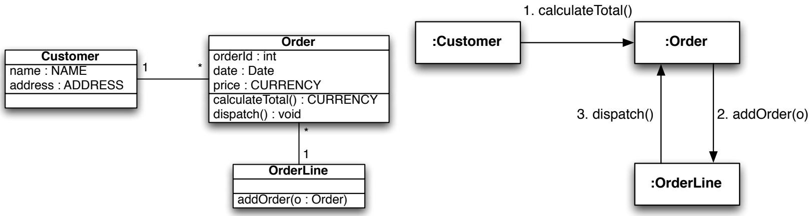
Inconsistency Example





Semi-Formal Methods

Inconsistency Example



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 $\longrightarrow$ 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

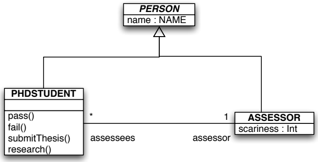
UML → B

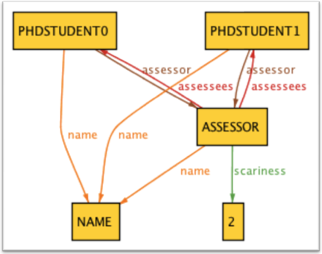
- **B** : Abstract 'Machines' [Abr96]
  - Focus on refinement towards an implementation
- U2B / Rodin UML-B [SB01, SB06]
  - Graphical formal language
  - Event-B semantics
  - Class-like diagrams

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





Tools/Methods

Comparison : Syntactic Completeness

Method/Tool	Syntax Covered
RoZ	Class
GeFoRME	Class, State, Object
AUtoZ	Class, State
Harmony	Class, State
UML $\longrightarrow$ Object-Z	Class
Ledang	Class, Collaboration, State, OCL
U2B	Class
Rodin UML-B	Class, State
UML2Alloy	Class, OCL

Tools/Methods

Comparison : Semantic Flexibility

Method/Tool	Semantic Flexibility
RoZ	Fixed / User-defined
GeFoRME	Flexible
AUtoZ	Flexible
Harmony	Fixed
UML $\rightarrow$ Object-Z	Fixed
Ledang	Fixed
U2B	Fixed
Rodin UML-B	Fixed
UML2Alloy	Fixed

Tools/Methods

Evaluation

- 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]

Conclusion

- 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

The End

- Thanks for listening
- Slides and bibliography available at http://www-users.cs.york.ac.uk/~jw
- Any questions?

References I

K. Anastasakis, B. Bordbar, G. Georg, and I. Ray.
UML2Alloy: A challenging model transformation.
In *ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems*, volume 4735, pages 436–450. Springer, 2007.

J.R. Abrial.
The B-book: assigning programs to meanings.
Cambridge University Press, 1996.

N Amalio.
Generative frameworks for rigorous model-driven development.
PhD thesis, Department of Computer Science, University of York, 2007.

P. Braun, H. Lotzbeyer, B. Schatz, and O. Slotosch.
Consistent integration of formal methods.
Lecture Notes in Computer Science, pages 48–62, 2000.

B. W. Boehm.
Software engineering: R & D trends and defense needs.
MIT Press, 1979.

E.M. Clarke and J.M. Wing.
Formal methods: State of the art and future directions.
ACM Computing Surveys (CSUR), 28(4):626–643, 1996.

S. Dascalu and P. Hitchcock.
An approach to integrating semi-formal and formal notations in software specification.
Proceedings of the 2002 ACM symposium on Applied computing, pages 1014–1020, 2002.

References II

S. Dupuy, Y. Ledru, and M. Chabre-Peccoud.
An overview of roz: a tool for integrating uml and z specifications.
Lecture Notes in Computer Science, pages 417–430, 2000.

D. Harel and B. Rumpe.
Meaningful modeling: what's the semantics of "semantics"?
Computer, 37(1010):64–72, 2004.

D Jackson.
Software Abstractions: Logic, Language and Analysis.
MIT Press, 2006.

S. Kim, D. Burger, and D. Carrington.
An MDA approach towards integrating formal and informal modeling languages.
Lecture Notes in Computer Science, 3582:448–464, 2005.

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.
Lecture Notes in Computer Science, 5287:381–393, 2008.

References III

H Ledang and J Souquires.
Integration of UML and B specification techniques: Systematic transformation from OCL expressions into B.
In *APSEC '02: Proceedings of the Ninth Asia-Pacific Software Engineering Conference*, pages 495–504. IEEE Computer Society, 2002.

International Standards Organisation.
Z formal specification notation : Syntax, type system and semantics.
International Standard, ISO/IEC 13568, 2002.

C. Snook and M. Butler.
Using a graphical design tool for formal specification.
pages 1–10, 2001.

C. Snook and M. Butler.
UML-B: Formal modeling and design aided by UML.
ACM Transactions on Software Engineering and Methodology (TOSEM), 15(1):92–122, 2006.

D.C. Schmidt.
Guest editor's introduction: Model-driven engineering.
IEEE Computer, 39(2):25–31, 2006.

L. T. Semmens, R. B. France, and T. W. G. Docker.
Integrated structured analysis and formal specification techniques.
The Computer Journal, 35(6):600–610, 1992.

References IV

I Sommerville.
Software Engineering.
Addison-Wesley, 6 edition, 2001.

E. Turner, H. Treharne, S. Schneider, and N. Evans.
Automatic generation of CSP—— B skeletons from xUML models, pages 364–379.
Springer, 2008.

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.
IEEE Computer, 23(9):8–24, 1990.

J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald.
Formal methods: Practice and experience.
ACM Computing Surveys, 41(4):1–36, 2009.

J.R Williams and F. Polack.
Automated formalisation for verification of diagrammatic models.
In *Proc. 6th International Workshop on Formal Aspects of Component Software*, Eindhoven, Netherlands, November 2009. Elsevier ENTCS.