A Survey of Formal Methods in Software Engineering

Dines Bjørner

DTU Informatics, Denmark

Univ. of Macau • APSEC 2012, Hong Kong
Structure of Talk

• An Example Formal Development 3
• What is Software? 7
• What is a Method? 8
• What is a Formal Method? 9
• History of Formal Method Specification Languages 10
• The Triptych Software Development Method 16
• Formal Methods in SE: State-of-Affairs 17
• Formal Methods in SE: Some Observations 23
• Closing 29
1. **An Example Formal Development**

1.1. **Fragments of A Domain Example**

1. A net (graph) consists of sets of links (arcs) and hubs (nodes).

2. Links and hubs have unique identifiers.

3. The mereology of links identifies two unique hubs.

4. The mereology of hubs identifies a set of hubs.

5. From a set of links one can extract its link identifiers.

6. From a set of hubs one can extract its hub identifiers.

7. Mereology identifiers identify existing net parts.

**type**

1. $N = L\text{-}set \times H\text{-}set$

2. $L\text{I}, H\text{I}$

**value**

2. $uid\_L\text{I}: L \to L\text{I}, uid\_H\text{I}: H \to H\text{I}$

3. $\text{mereo}\_L: L \to HI\text{-}set$

4. $\text{mereo}\_H: H \to LI\text{-}set$

5. $xtr\_L\text{Is}: L\text{-}set \to LI\text{-}set$

6. $xtr\_H\text{Is}: H\text{-}set \to HI\text{-}set$

**axiom**

7. $\forall (ls,hs):N \cdot$

3. $\forall l:L \cdot l \in ls \Rightarrow \text{card } \text{mereo}\_L(l) = 2 \land$

7. $\text{mereo}\_L(l) \subseteq xtr\_H\text{Is}(hs) \land$

7. $\forall h:H \cdot h \in hs \Rightarrow$

7. $\text{mereo}\_H(h) \subseteq xtr\_L\text{Is}(ls)$
• The above models general nets, see left figure below.

Figure 1: General Net and Toll-road Net

• Next we model toll-road nets, see right figure above.
1.2. Fragments of A Requirements Example

1.2.1. Net Instantiation

8. A toll-road system consists of \( n \) toll-road segments and \( n + 1 \) triples of toll plaza connections.

9. A toll-road segment is a pair of opposite traffic-direction toll roads.

10. A toll plaza connection consists of a toll plaza hub, a plaza-to-toll-road link and a toll-road hub.

\[
\text{type} \\
\text{8. TRS} = \text{TRS}^{*} \times \text{TPC}^{*} \\
\text{axiom} \\
\text{8. } \forall (lll,hlh):\text{TRS} \cdot \text{len } hlh = \text{len } lll + 1
\]

\[
\text{type} \\
\text{9. TRS} = L \times L \\
\text{10. TPC} = H \times L \times H
\]

1.2.2. Net Abstraction

Toll-road systems are concrete instantiations of nets.

11. We therefore define a net abstraction function

12. which from toll-road systems

13. abstracts nets.

\[
\text{value} \\
\text{11. abs}_N: \text{TRS} \rightarrow N \\
\text{12. } \text{abs}_N(\text{trs},\text{tpc}) \equiv \\
\text{13. } \{\{\text{lf,lt}\}|(\text{lf,lt}) : \text{TRS} \cdot (\text{lf,lt}) \in \text{elems trs}\} \\
\text{13. } \cup \{l|L\cdot(\_\_\_\_) \in \text{elems tpcl}\}, \\
\text{13. } \{\{\text{hp,ht}\}|(\text{hp,ht}) : \text{TPC} \cdot (\text{hp,ht}) \in \text{elems tpcl}\}
\]
1.3. **Fragments of A Software Design Example**

- We decide to implement the toll-road net
- as a collection of relational database relations.

14. The transport net relational database consists of five relations.
   - a relation for hub mereologies,
   - a relation for hub attributes,
   - a relation for link mereologies,
   - a relation for link locations, and
   - a relation for other link attributes.

15. For a given hub (hi:HI) there is any set of mereology tuples.

16. For a given hub there is one other attributes tuple.

17. For a given link there is one mereology tuple.

18. For a given link there is a set of at least two location tuples (ll:LL).

19. For a given link there is one other attributes tuple.

**type**

15. HM = HI × LI
16. HA = HI × LOC × ... 
17. LM = LI × HI × HI
18. LL = LI × LOC
19. LA = LI × LEN × ...
2. What is Software?

- **Software.** By software we shall understand all the following kinds of documents:

  - **Planning Docs.**
    - Background
    - Motivation
    - Teams
    - Etcetera.

  - **Development Docs.**
    - Domain description
    - Requirements prescription
    - Software design & code
    - Test data and results

  - **Model checking**
  - **Proof of properties**

  - **Manuals**
    - Installation
    - Education
    - Maintenance
    - etcetera

  - **Project Docs.**
    - Planning, Budget, Accounts
    - Project Logs
3. What is a Method?

- **Method.** By a *method* we shall understand
  - a set of **principles**
  - for **selecting** and **applying**
  - a number of **techniques** and **tools**
  - in order to **analyse** and **synthesize** (construct) an **artifact**.

- Example **tools**: specification and coding languages, theorem provers, model checkers, test tools, etc.
- Example **techniques**: abstract and concretisation, proof techniques, etc., refinement, etc.
- Example **analyses**: consistency, completeness, invariants, etc.
4. What is a Formal Method?

- **Formal Method.** By a *formal method* we shall understand
  - a **comprehensive** set of method techniques and tools
  - which have a **formal foundation in mathematics,**
  - that is:
    - each specification language has
      - a *mathematical syntax,*
      - a *mathematical semantics,* and
      - a *proof system;*
    - while supporting
      - *refinement,*
      - *proof,*
      - *model checking,*
      - *test,*
      - etcetera, tools obey these formalisms.
5. **History of Formal Method Specification Languages**

- A selection of basically model-oriented methods:
  - VDM
  - Z
  - RAISE
  - B, Event B
  - Alloy

- Other formal methods are property-oriented:
  - CafeOBJ,
  - CASL,
  - Maude, etc.
5.1. VDM

- PL/I Compiler Devt. P. Lucas, H. Bekič (†), C.B.Jones and D.Bjørner
- Springer LNCS 61 1978 and Prentice-Hall 1982

- VDM SL (Spec.Lang.) Standard, 1996: ISO/IEC 13817/1
- http://www.vdmportal.org/twiki/bin/view
- Lively VDM activity in Japan and Europe: Research and Industry
5.2. **Z**

- Z is developed by **Jean-Raymond Abrial** between 1980–1990.
- Lively research around Z in mostly England (Woodcock, Univ. of York)

- Major british industrial uses of Z:
  - etcetera ...

- **Z Standard** ISO/IEC 13568, 2002
5.3. RAISE

- RAISE: Rigorous Approach to Industrial Software Engineering
- Result of an EU ESPRIT BRA project with
  DDC: Dansk Datamatik Center (Bjørner: Instigator) and
  STL: Standard Telephone Labs., UK, etc. 1985–1990
- RAISE is being used at Terma Space Division, a Danish Systems house.
- RSL (RAISE Spec.Lang.) captures concurrency and features Duration Calculus
- RAISE was the formal method being used at UNU-IIST, Macau, 1992–2009
  - Chinese Railways
  - Vietnam Ministry of Finance
  - Philippine Min. of Telecomm.
  - Chennai Harbour Management, India

- Primarily designed by Søren Prehn and Chris George. I am using it!
5.4. B, Event B

- B was developed by **Jean-Raymond Abrial** between 1990--2000.
- Event-B is developed by **Jean-Raymond Abrial** since 2000.
- Event-B evolved from a rather total redesign of B.
- Event-B captures a form of concurrency.
- [http://www.event-b.org/](http://www.event-b.org/)
- French B and Event-B industrial users.
- Academic base in France (Nancy) and the UK (Southampton)
5. History of Formal Method Specification Languages

5.5. Alloy

- Masterminded by Daniel Jackson
- An elegant VDM “derivative”
- Great for teaching abstraction and formal methods.
- My strongest recommendation for introduction for formal methods.
6. The Triptych Software Development Model

6.1. The Dogma

- Before software can be designed (i.e., coded, programmed)
- one must have a reasonable understanding of its requirements.
- Before requirements can be prescribed
- one must have a reasonable understanding of their domain.

6.2. Consequences of the Dogma

- Thus software engineering has three major development phases:
  - **domain** engineering: resulting in a **description**, \( D \)
  - **requirements** engineering: resulting in a **prescription**, and \( R \)
  - **software design**, \( S \)

\[ D, S \models R \]
7. Formal Methods: State-of-Affairs

7.1. History

- First industry scale formal developments were the DDC CHILL and Ada compiler developments: 1980–1984

Figure 2: CHILL and Ada Software Development Graphs
7.2. **Industrial Uptake**

- Slow, but steady

### 7.2.1. Software Industries

- Denmark: Terma
- England: Altran-Praxis
- France: ClearSy
- Germany: Verified Sys.
- Italy: Ansaldo
- Japan: CSK
- Netherlands: CHESS
- Sweden: Telelogic (IBM)
- Russia: ISP/RAS
- USA: SRI, Microsoft

### 7.2.2. Hardware Industries: Verified Chip Designs

- Intel
- AMD
- Cadence Berkeley
- IBM
7.2.3. More FM URLs

  DEPLOY Success Stories
  http://www.fm4industry.org/index.php/DEPLOY_Success_Stories

- US DoD NASA: Langley Formal Methods
  http://shemesh.larc.nasa.gov/fm/

- SRI Inc., Computer Systems Lab.
  http://www.csl.sri.com/programs/formalmethods/

- Laboratory for Reliable Software (LaRS)
  http://lars-lab.jpl.nasa.gov/

- Altran-Praxis: Formal Computing

- ClearSy B Method
  http://www.clearsy.com/our-specific-know-how/b-method/?lang=en

- Formal Methods Wiki
  http://formalmethods.wikia.com/wiki/Formal_Methods_Wiki
7.3. **Industrial Needs**

- Industries that are using FMs on projects
  - need all SEs on that project to have learned one or another of the methods listed earlier;
  - it will not work with any mixture of professional and non-professional SEs;
  - these software houses need a steady — local — supply of such professionally trained SEs.
7.4. University Courses

7.4.1. BSc Courses

- Functional Programming
- Imperative Programming
- Logic Programming
- Parallel Programming
- Abstraction and Modelling

Standard ML
Spec #
Prolog
CSP (as in e.g. Java)
See [1]

[1] or [1]
7.4.2. MSc Courses

- Languages and Systems
- Domains, Requirements, Software Design
- Advanced Software Verification: Formal Testing, Model Checking, Theorem Proofs

See [2] and [3]

8.1. Formal Methods and Formal Techniques

- By **formal methods** software development we mean
  - a development which uses formal specification languages
  - in all there phases of development: domains, requirements and design

- By **formal techniques** software development we mean
  - a development which uses one or another formal techniques
  - usually design only —
  - these formal techniques could be
    - static analysis,
    - formal testing,
    - model checking,
    - theorem proving.
8.2. From Mono-language to Multi-language Specification

- The VDM-SL, Z, B/Event B and Alloy Spec. Langs. are OK —
  - but they cannot cope with one or another facet of software,
  - so their use must be accompanied by use of
    - CSP,
    - Petri Nets,
    - Temporal Logic,
    - MSC,
    - State Charts,
    - etcetera,
- CSP and DC (Duration Calculus) can be used with RSL.
8.3. Sociology of Acceptance of Formal Methods

8.3.1. Industry

- The software (SW) industry has been moderately successful
  - COTS\(^1\) SW in particular (MS, etc.),
  - but Turn-key SW projects have failed on a gigantic scale,
  - yet the SW industry persists in believing
  - that such projects can be staffed by non-professionals.
- The SW industry, in general, resists FMs
  - claiming that there are no statistics supporting FMs:
    - there are such “statistics”,
    - but real such requires at least a triplet of 1000 test devts.
- And: what would they do with all their non-professional SEs?

\(^1\)COTS: Commercial off-the-shelf
8.3.2. Universities

- In a mathematics dept. all mathematicians
  - know enough of colleagues’ specialised field,
  - to appreciate it, and “interface” to, i.e., make use of it.
- In most computer science depts. such is not the case:
  - so-called theoretical CSs do not know how to develop software,
  - let alone of the kind of FMs covered in this talk.
- Their students, consequently, do not take FMs serious.
8.4. Inevitability of FMs

- The MS *Distributed File System Replication* DFS R “Story”\(^2\)
  - Microsoft is increasingly committed to Formal Techniques

- If software can have guaranteed warranties (‘correctness’),
  - then that will occur
  - and software development will hence use FMs.

- As soon as customers discover the possibility of certified software
  - then they will demand it
  - and only software developed using FMs rigorously can offer that.

8.5. Textbooks


- **RAISE**: D. Bjørner. Software Engineering,
  Vol.1: *Abstraction and Modelling*,
  Vol.2: *Specification of Systems and Languages*,
  Vol.3: *Domains, Requirements and Software Design*.


9. Closing

• This has been a “lightweight” survey of formal methods and industry.

• This was deliberately so —
  ⨁ so that you can ask questions
  ⨁ and I can hopefully answer them;
  ⨁ at least we can discuss the state-of-affairs.

Many Thanks — and: Questions?