

# 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

- |   |   |
|---|---|
| <ol style="list-style-type: none"> <li>1. A net (graph) consists of sets of links (arcs) and hubs (nodes).</li> <li>2. Links and hubs have unique identifiers.</li> <li>3. The mereology of links identifies two unique hubs.</li> <li>4. The mereology of hubs identifies a set of hubs.</li> <li>5. From a set of links one can extract its link identifiers.</li> <li>6. From a set of hubs one can extract its hub identifiers.</li> <li>7. Mereology identifiers identify existing net parts.</li> </ol> | <p><b>type</b></p> <ol style="list-style-type: none"> <li>1. <math>N = L\text{-set} \times H\text{-set}</math></li> <li>2. LI, HI</li> </ol> <p><b>value</b></p> <ol style="list-style-type: none"> <li>2. <math>\text{uid\_LI}: L \rightarrow \text{LI}</math>, <math>\text{uid\_HI}: H \rightarrow \text{HI}</math></li> <li>3. <math>\text{mereo\_L}: L \rightarrow \text{HI-set}</math></li> <li>4. <math>\text{mereo\_H}: H \rightarrow \text{LI-set}</math></li> <li>5. <math>\text{xtr\_LIs}: L\text{-set} \rightarrow \text{LI-set}</math></li> <li>6. <math>\text{xtr\_HIs}: H\text{-set} \rightarrow \text{HI-set}</math></li> </ol> <p><b>axiom</b></p> <ol style="list-style-type: none"> <li>7. <math>\forall (ls, hs): N \cdot</math></li> <li>3. <math>\forall l: L \cdot l \in ls \Rightarrow \mathbf{card} \text{mereo\_L}(l) = 2 \wedge</math></li> <li>7. <math>\text{mereo\_L}(l) \subseteq \text{xtr\_HIs}(hs) \wedge</math></li> <li>7. <math>\forall h: H \cdot h \in hs \Rightarrow</math></li> <li>7. <math>\text{mereo\_H}(h) \subseteq \text{xtr\_LIs}(ls)</math></li> </ol> |
|---|---|

- The above models general nets, see left figure below.

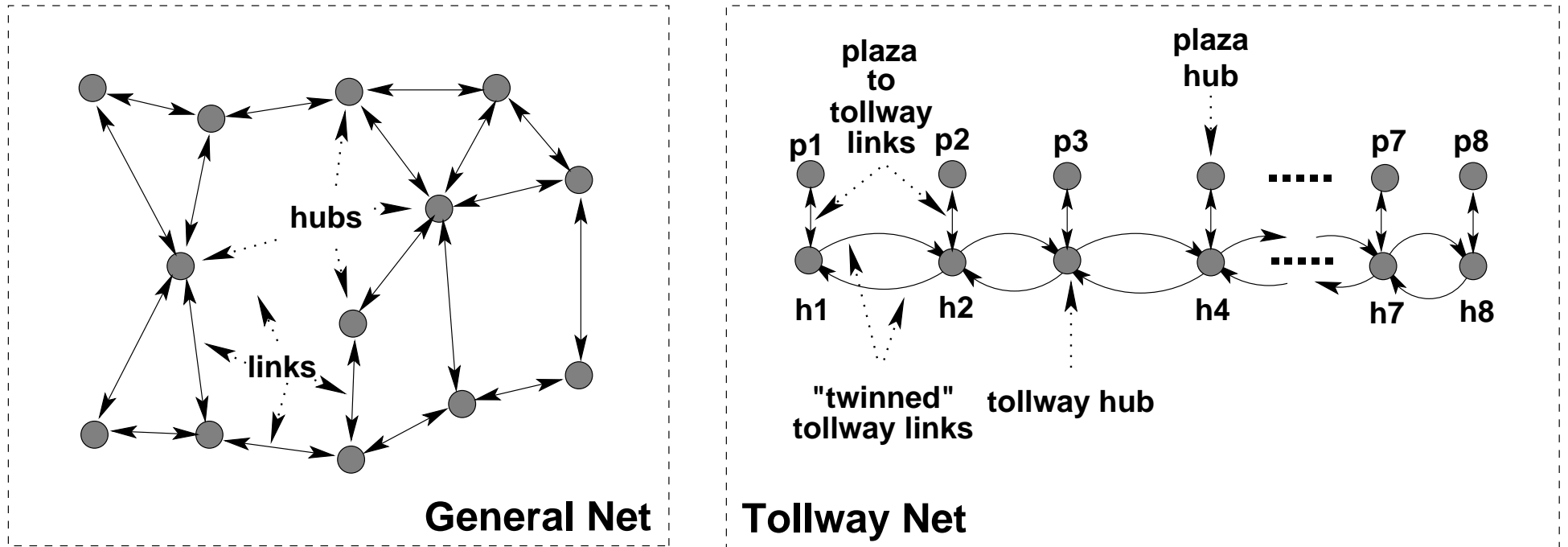


Figure 1: General Net and Toll-road Net

- 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.

**type**

8.  $\text{TRS} = \text{TRS}^* \times \text{TPC}^*$

**axiom**

8.  $\forall (\text{lll}, \text{hlh}) : \text{TRS} \cdot \mathbf{len} \text{ hlh} = \mathbf{len} \text{ lll} + 1$

**type**

9.  $\text{TRS} = \text{L} \times \text{L}$

10.  $\text{TPC} = \text{H} \times \text{L} \times \text{H}$

### 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.

**value**

11.  $\text{abs\_N} : \text{TRS} \rightarrow \text{N}$

12.  $\text{abs\_N}(\text{trsl}, \text{tpcl}) \equiv$

13.  $(\{\{lf, lt\} \mid (lf, lt) : \text{TRS} \cdot (lf, lt) \in \mathbf{elems} \text{ trsl}\}$

13.  $\cup \{l \mid l : \text{L} \cdot (\_, l, \_) \in \mathbf{elems} \text{ tpcl}\},$

13.  $\{\{hp, ht\} \mid (hp, \_, ht) : \text{TPC} \cdot (hp, \_, ht) \in \mathbf{elems} \text{ tpcl}\})$

## 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**
14.  $RN = HM\text{-set} \times HA\text{-set} \times LM\text{-set} \times LL\text{-set} \times LA\text{-set}$
15.  $HM = HI \times LI$
16.  $HA = HI \times LOC \times \dots$
17.  $LM = LI \times HI \times HI$
18.  $LL = LI \times LOC$
19.  $LA = LI \times LEN \times \dots$

## 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:

● <b>VDM</b>	11
● <b>Z</b>	12
● <b>RAISE</b>	13
● <b>B, Event B</b>	14
● <b>Alloy</b>	15

- Other formal methods are property-oriented:

- ❖ **CafeOBJ,**
- ❖ **CASL,**
- ❖ **Maude,** etc.

## 5.1. VDM

- VDM: [IBM] Vienna [laboratory software] Development Method 1973 – 1975
- PL/I Compiler Devt. **P. Lucas, H. Bekič** (†), **C.B.Jones** and **D.Bjørner**
- Springer LNCS 61 1978 and Prentice-Hall 1982



Bjørner and Jones

- Dansk Datamatik Centre: CHILL (CCITT) and Ada (US DoD) Language Definitions and Compiler Devts 1981–1984. DDCI Inc., USA
- VDM SL (Spec.Lang.) Standard, 1996: ISO/IEC 13817/1
- VDM Tools: JFITS, CSK, Japan: <http://www.vdmbook.com/download.html>
- <http://www.vdmportal.org/twiki/bin/view>
- Lively VDM activity in Japan and Europe: Research and Industry

## 5.2. Z

- Z for Zermelo (1871–1953) – Fraenkel (1891–1965) Set Theory
- 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:
  - ❖ **Altran-Praxis** <http://www.altran-praxis.com/>
  - ❖ etcetera ...
- [http://formalmethods.wikia.com/wiki/Z\\_User\\_Group](http://formalmethods.wikia.com/wiki/Z_User_Group)
- **Z Standard** ISO/IEC 13568, 2002

## 5.3. RAISE

- RAISE: **R**igorous **A**pproach to **I**ndustrial **S**oftware **E**ngineering
- 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

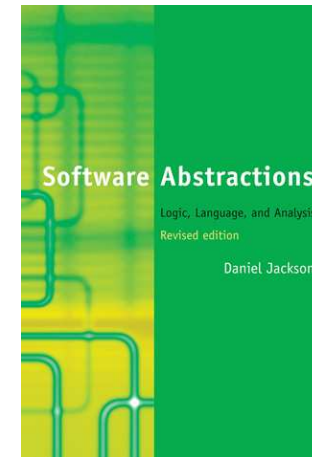


- Primarily designed by **Søren Prehn** and **Chris George**. I am using it !

## 5.4. B, Event B

- B for **Bourbaki**: Collective pseudonym author name of mathematics monographs: [http://www.en.wikipedia.org/wiki/Nicolas\\_Bourbaki](http://www.en.wikipedia.org/wiki/Nicolas_Bourbaki)
- 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/>
- French B and Event-B industrial users.
- Academic base in France (Nancy) and the UK (Southampton)

## 5.5. Alloy



- Masterminded by **Daniel Jackson**
- An elegant VDM “derivative”
- <http://alloy.mit.edu/alloy/>
- 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**,
  - ❖ **requirements** engineering: resulting in a **prescription**, and
  - ❖ **software design**,

*D**R**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**

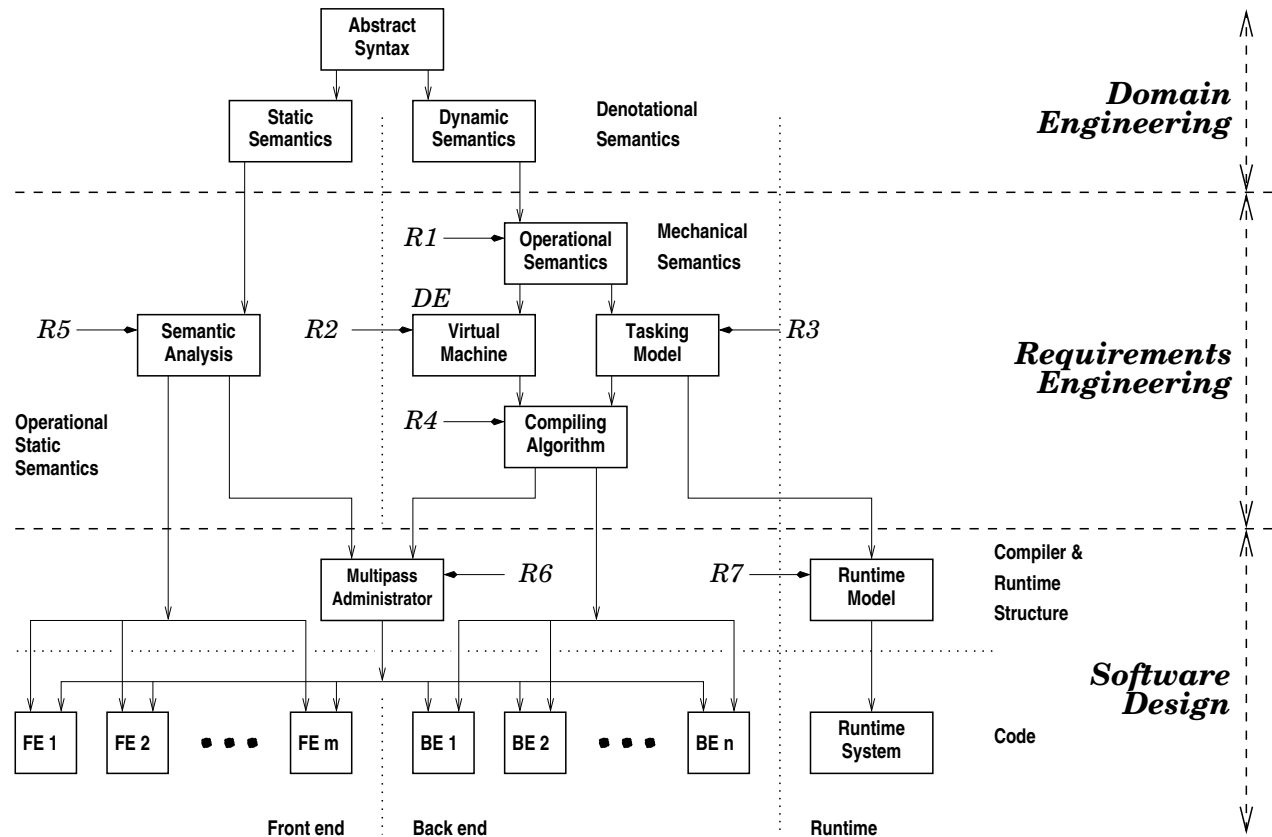


Figure 2: CHILL and Ada Software Development Graphs

## 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: CHESSE
- 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

- ERCIM FMICS: Europ.Res.Cons. Industrial Critical Systems  
DEPLOY Success Stories  
[http://www.fm4industry.org/index.php/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  
<http://www.altran-praxis.com/formalComputing.aspx>
- 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](http://formalmethods.wikia.com/wiki/Formal_Methods_Wiki)

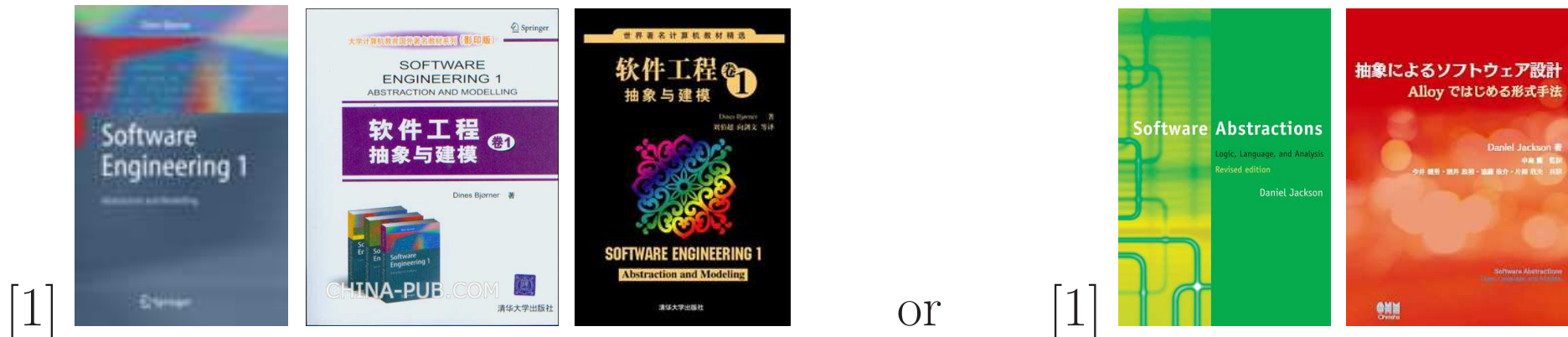
## 7.3. Industrial Needs

- 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 Standard ML
- Imperative Programming Spec #
- Logic Programming Prolog
- Parallel Programming CSP (as in e.g. Java)
- Abstraction and Modelling See [1]



## 7.4.2. MSc Courses

- Languages and Systems See [2]
- Domains, Requirements, Software Design See [3]



- Advanced Software Verification:  
Formal Testing, Model Checking, Theorem Proofs

## 8. Formal Methods: Some Observations

### 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,
    - ⊗ MSC,
    - ⊗ Petri Nets,
    - ⊗ State Charts,
    - ⊗ Temporal Logic,
    - ⊗ 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<sup>1</sup> 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 ?

---

<sup>1</sup>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”<sup>2</sup>
  - ❖ 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.

---

<sup>2</sup><http://research.microsoft.com/pubs/70451/tr-2007-75.pdf>

## 8.5. Textbooks

- **VDM:** J. Fitzgerald and P. G. Larsen. *Modelling Systems – Practical Tools and Techniques in Software Development*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 1998.
- **Z:** J. C. P. Woodcock and J. Davies. *Using Z: Specification, Proof and Refinement*. Prentice Hall International Series in Computer Science, 1996.
- **RAISE:** D. Bjørner. Software Engineering,  
Vol.1: *Abstraction and Modelling*,  
Vol.2: *Specification of Systems and Languages*,  
Vol.3: *Domains, Requirements and Software Design*.  
Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.
- **B, Event B:** J.-R. Abrial.  
*The B Book: Assigning Programs to Meanings and  
Modeling in Event-B: System and Software Engineering*.  
Cambridge University Press, Cambridge, England, 1996 and 2009
- **Alloy:** D. Jackson. *Software Abstractions: Logic, Language, and Analysis*.  
The MIT Press, Cambridge, Mass., USA, April 2006. ISBN 0-262-10114-9.

## 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 ?**