Formal Methods

and its Relevance for Industry and Emmergent Markets

Prof. Burkhart Wolff
Univ - Paris-Sud / LRI

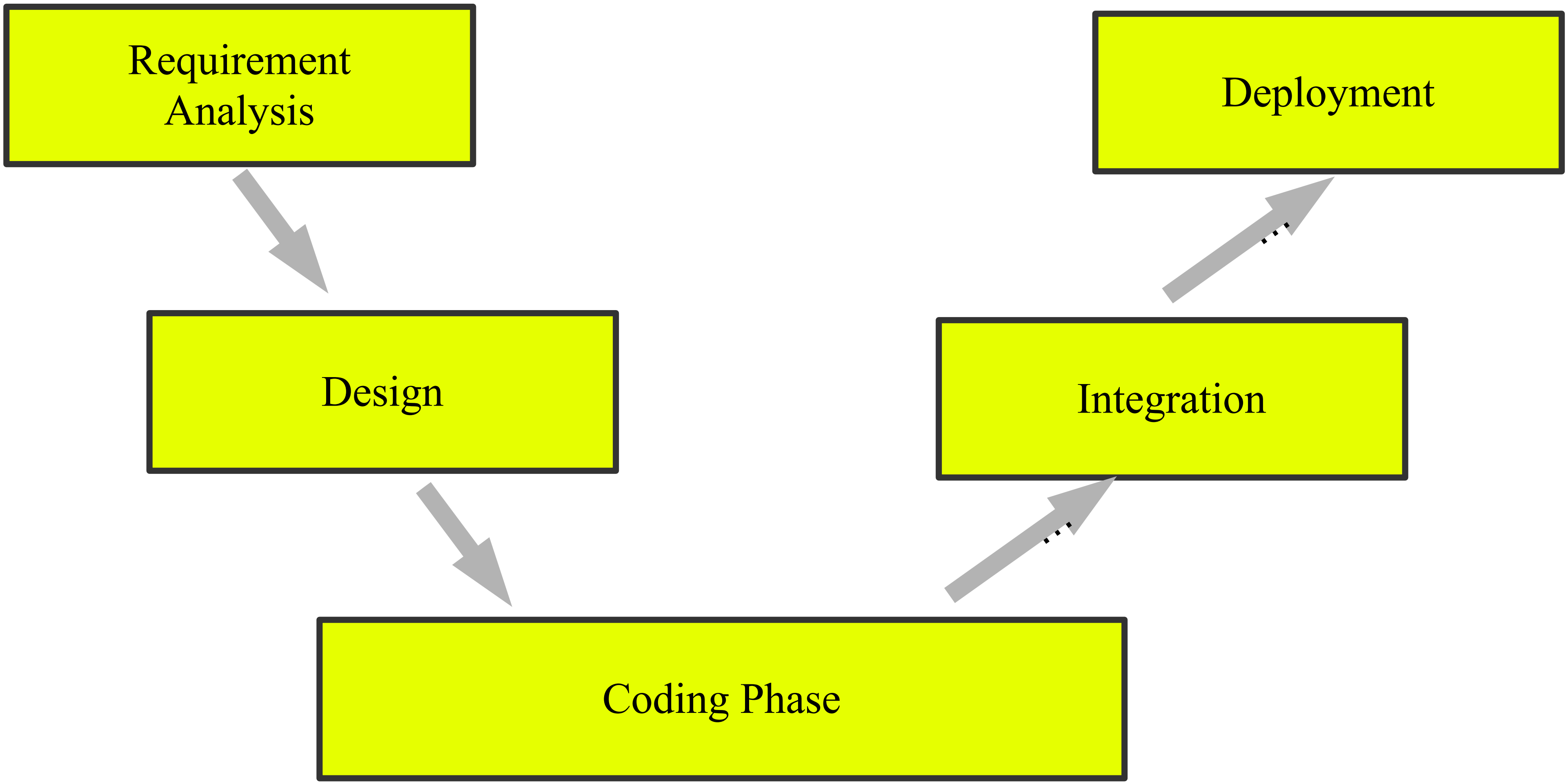
Burkhart Wolff

Université Paris-Sud

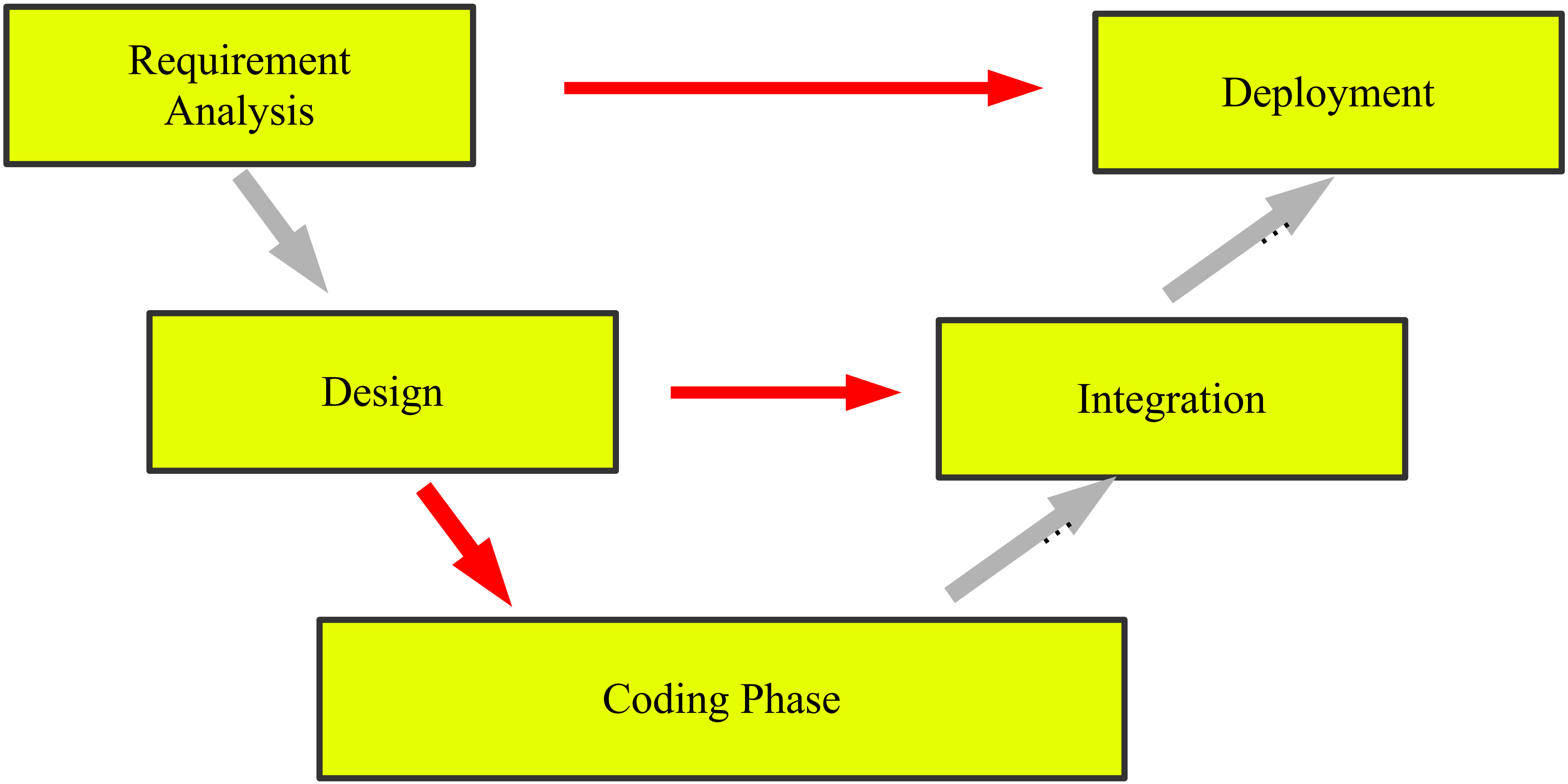
Sort of an Introduction

- We have seen a lot of discussions these days what communication systems are built nowadays ...
- I'd like to shift the question on
  - how were (high quality) systems were built ?
  - what are the necessary processes and tools ?
  - how can engineers in a project detect that the right system is built ?
  - how can engineers in a project detect the the system is built right ?

Why is so difficult to get software right?



Why is so difficult to get software right?



Sort of an Introduction

- These are problems adressed by sub-field in software engineering, called Formal Methods. They:
  - … have their roots in Formal Logic and Math
  - … are fundamental for Program Analysis and Automated Program Construction
  - … are nowadays key-technology for systems
    - complex
    - mission, safety- or security critical
      - for which legislative or certification procedures require this …

Formal Methods Today - Outline

- Brief History
- TOP-DOWN: Model-Driven Approaches ("Correctness by Construction")
- BOTTOM-UP: Approaches like Code-Verification/Verifying Compilers
- Relevance in Industrial Applications Today
- Relevance for Emmerging Countries ?
- A Perspective for Teaching at ICT - IIT Rajastan

Brief History

- early approaches to automated theorem proving (ATP):
  - Turing 52 !!!
  - Nelson / Oppen 60
  - Robinson Resolution Procedure : 62
- Problem is fundamentally hard: decidability of PL is NP,
  (nearly all approaches in ATP suffer from
  state explosion - still today ....)
  FOL is undecidable, HOL is even incomplete ...

Brief History

- Hoare Calculus (BOTTOM UP) 1972
  Dijkstra/Floyds WP (BOTTOM UP) 1976
- Algebraic Specification (TOP DOWN) 1980
  Refinement Calculus (TOP DOWN) 1990
- Z [86], B[90], CSP[86], CCS[88], ...
- Interactive Theorem Prover:
  Edinburg LCF [82], Coq [86], Isabelle [86]
  Automated TP: Otter, . . ., BoyerMoore 78
- Abstract Interpretation (TOP DOWN)
  Cousot~[80], HankinBurnAbramski [86]

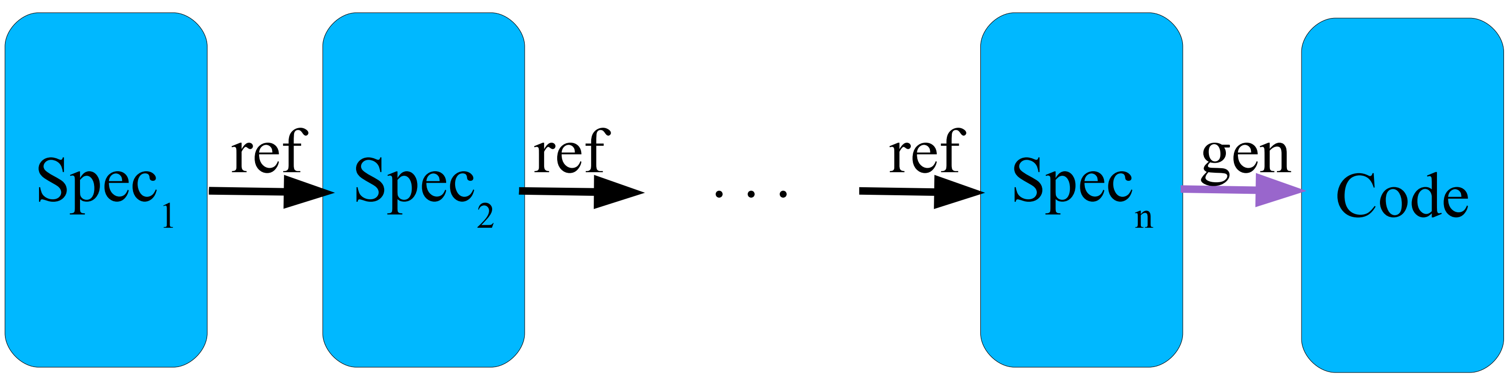
Brief History

- …
- UML / OCL [03 …] started as informal lang.
- ESC Java [04] (with ATP Simplify), Spec#[08]
- Automated Provers: AltErgo, Z3 **[08]**
- Verifying Compilers for C: Isabelle/Simpl,[06] VCC[08], Frama-C/Jessie[07] etc.
- Test: Korat [02], SpecExplorer[05], Pexx[06]
- Refinement: Rhodin System [08]

…

Model-Driven Approaches ("Correctness by Construction", MDE)

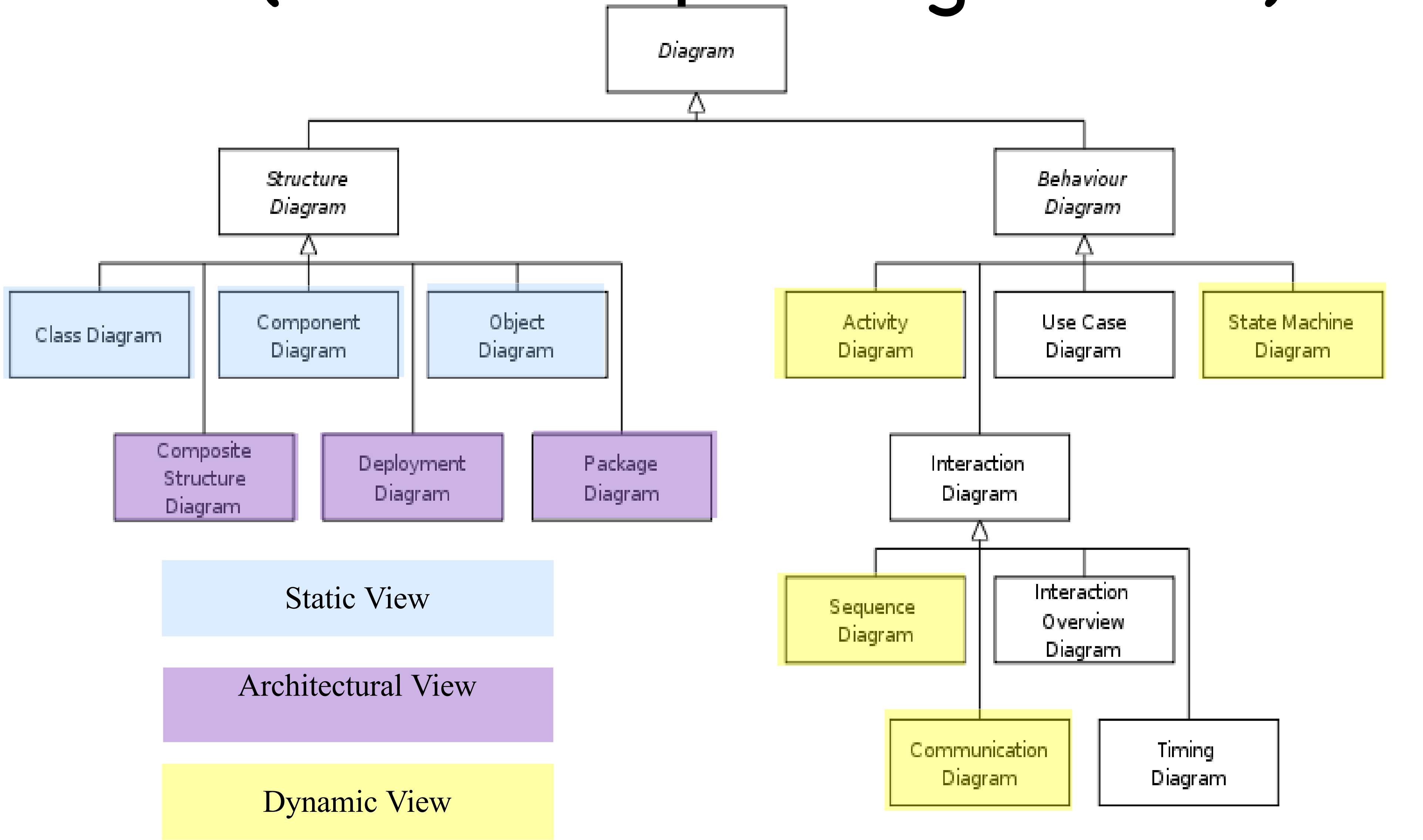
- Refinement / Transformation Oriented Approaches: writing a model, refine it to concreter models,generate code (Z, CSP, B, Refinement Calculus, UML xxx)

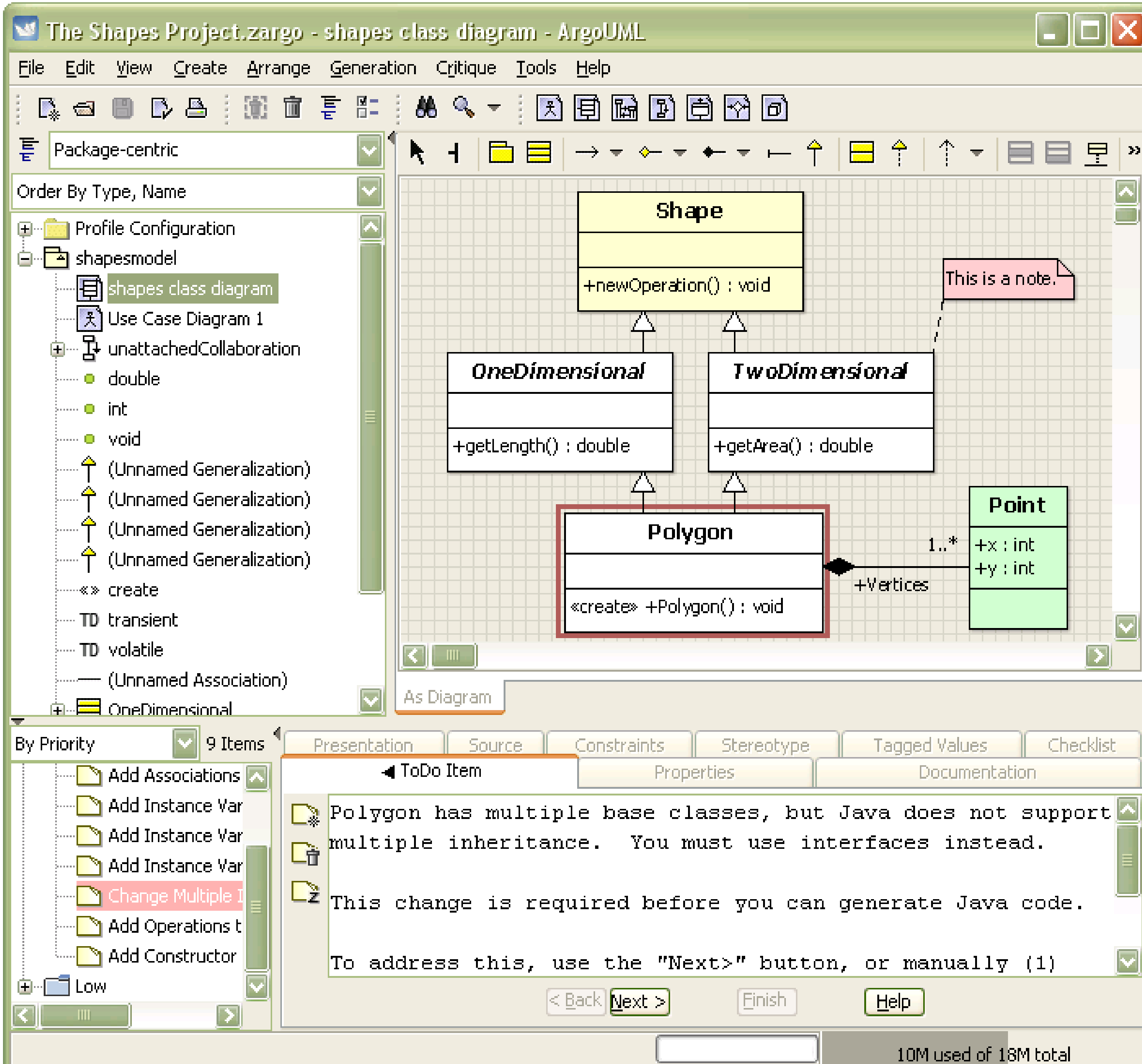


Example TOP-DOWN : The UML
it offers the advantage ...

- ... of being a basis for Integrated Development Environments (IDE's like ArgoUML, Poseidon, Rational Rose, ...)
- ... to offer „object-oriented" specifications in form of pre- and post conditions + behaviour descriptions
- ... to offer a formal, mathematical semantics (well, at least to some parts of the UML)
- ... to be fairly widely used in industry, even if not always supported entirely
- ... is the basis for a whole software-engineering paradigm called Model-Driven Engineering (MDE).

The UML 2.0 Diagrams (for corresponding models)




The Shapes Project.zargo - shapes class diagram - ArgoUML
File Edit View Create Arrange Generation Critique Tools Help
Package-centric
Order By Type, Name
Profile Configuration
shapesmodel
shapes class diagram
Use Case Diagram 1
unattachedCollaboration
double
int
void
(Unnamed Generalization)
(Unnamed Generalization)
(Unnamed Generalization)
(Unnamed Generalization)
create
transient
volatile
(Unnamed Association)
OneDimensional
Shape
+newOperation() : void
This is a note.
OneDimensional
+getLength() : double
TwoDimensional
+getArea() : double
Point
+x : int
+y : int
1..*
+Vertices
Polygon
«create» +Polygon() : void
As Diagram
By Priority
9 Items
Add Associations
Add Instance Var
Add Instance Var
Add Instance Var
Change Multiple I
Add Operations t
Add Constructor
Low
Presentation
Source
Constraints
Stereotype
Tagged Values
Checklist
ToDo Item
Properties
Documentation
Polygon has multiple base classes, but Java does not support multiple inheritance. You must use interfaces instead.
This change is required before you can generate Java code.
To address this, use the "Next>" button, or manually (1)
< Back
Next >
Finish
Help
10M used of 18M total

The HOL-OCL Environment

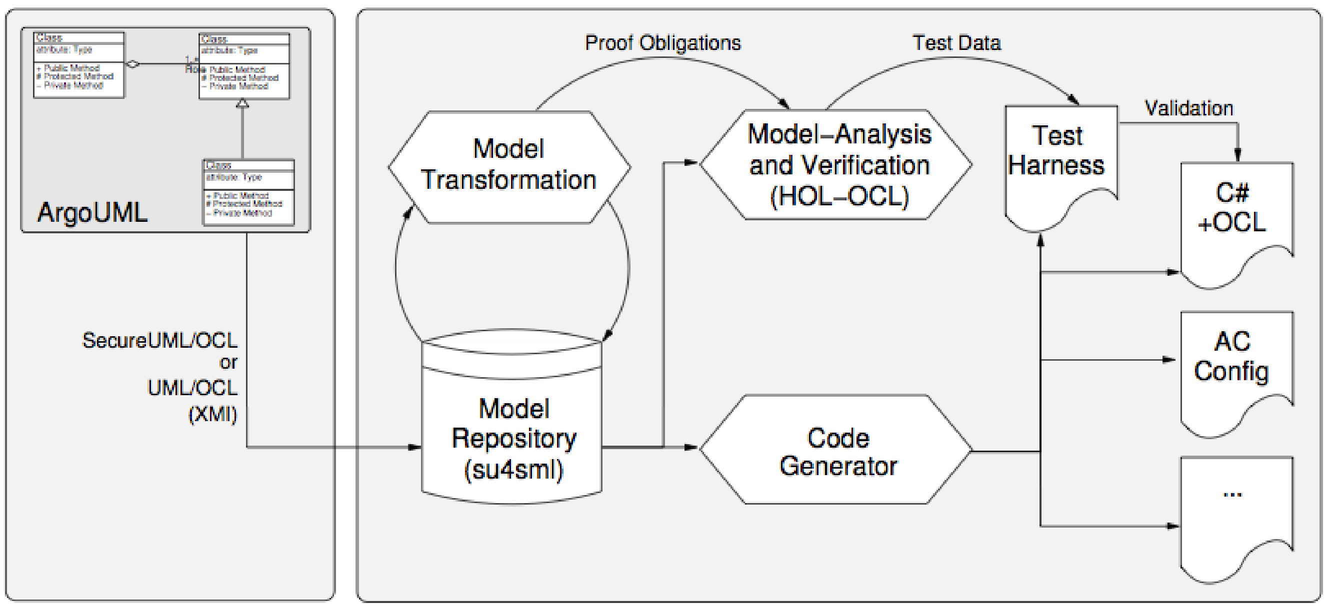
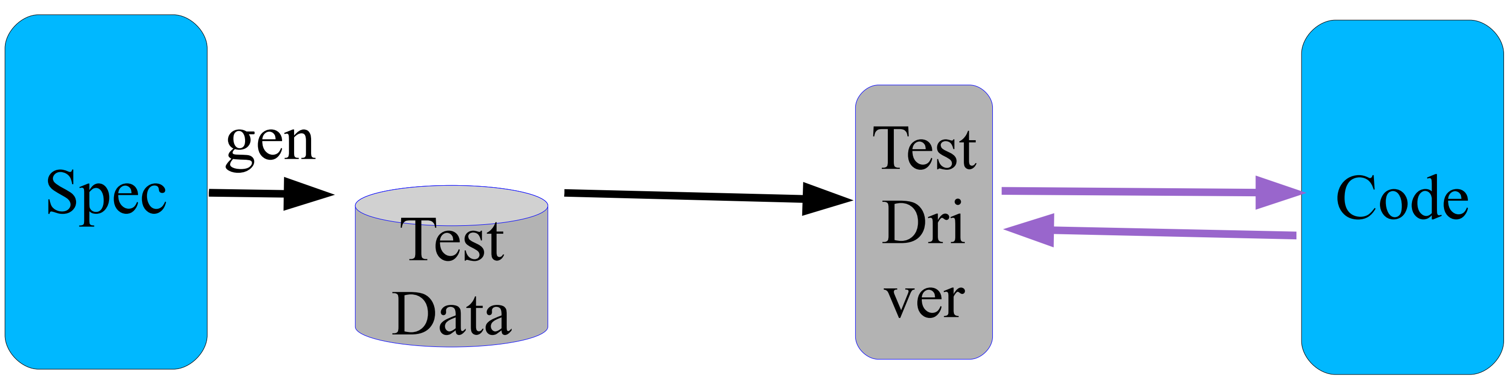


Figure 1: MDA Framework and Toolchain Overview

Model-Driven Approaches (Model-Based Testing (MBT))

- Test-Generation Oriented Approaches: writing a model, writing a program, generate Test Cases to check conformance (Z, Pexx, SpecExplorer, HOL-TestGen)



Model-based Testing …

- … can be done post-hoc; significant industrial projects "reverse engeneer" legacy system models
- … attempts to find bugs in specifications EARLY (and can complement verification projects …)
- … can help system integration processes by assuring that third-party components are in fact usable in a larger system.

  The model gets the role of a "contract" in this scenario.

Our System: HOL-TestGen is ...

- ... based on HOL (Higher-order Logic):
  - "Functional Programming Language with Quantifiers"
  - plus definitional libraries on Sets, Lists, . . .
  - can be used meta-language for HoareCalculi, Z, CSP. . .
- ... implemented on top of Isabelle
  - an interactive prover implementing HOL
  - the test-engineer must decide over, abstraction level, split rules, breadth and depth of data structure exploration . . .
  - providing automated and interactive constraint-resolution techniques
  - interface: ProofGeneral
- ... by thy way, a verified test-tool

HOL-TestGen Workflow

- Modelisation
  - writing background theory of problem domain

HOL-TestGen Workflow

- Modelisation
  - writing background theory of problem domain
- Test-Case-Generation from Test-Specification
  - automated procedure gen_test_case ...
  - Test-Cases: partitions of I/O relation of the form

$$C_1(x) \Longrightarrow \ldots\ C_n(x) \Longrightarrow \text{post } x\ (\text{PUT } x)$$

HOL-TestGen Workflow

- Modelisation
  - writing background theory of problem domain
- Test-Case-Generation from Test-Specification
  - automated procedure gen_test_case …
  - Test-Cases: partitions of I/O relation of the form

$$C_1(x) \Longrightarrow \ldots C_n(x) \Longrightarrow \text{post } x \text{ (PUT } x)$$

- Test-Data-Selection
  - constraint Solver gen_test_data
  - finds x satisfying $C_i(x)$

HOL-TestGen Workflow

- Modelisation
  - writing background theory of problem domain
- Test-Case-Generation from Test-Specification
  - automated procedure gen_test_case ...
  - Test-Cases: partitions of I/O relation of the form

$$C_1(x) \Longrightarrow \ldots\ C_n(x) \Longrightarrow \text{post } x \text{ (PUT } x)$$

- Test-Data-Selection
  - constraint solver gen_test_data
  - finds x satisfying $C_i(x)$
- Test-Driver Generation
  - automatically compiled, drives external program

HOL-TestGen Workflow

- Modelisation
  - writing background theory of problem domain
- Test-Case-Generation from Test-Specification
  - automated procedure gen_test_case ...
  - Test-Cases: partitions of I/O relation of the form

$$C_1(x) \Longrightarrow \ldots C_n(x) \Longrightarrow \text{post } x \ (\text{PUT } x)$$

- Test-Data-Selection
  - constraint solver gen_test_data
  - finds x satisfying $C_i(x)$
- Test-Driver Generation
  - automatically compiled, drives external program
- Test Execution, Test-Documentation

Mini-Example

- Modelisation
  - is_sorted, insert, sort
- Test-Case-Generation from Test-Specification
  - Test-Sepcification: sort x = PUT x
  - Test-Cases: $\ldots\ x \leqslant y \Longrightarrow [x,y] = \mathrm{PUT}\ [x,y]$

    $\ldots\ x > y \Longrightarrow [y,x] = \mathrm{PUT}\ [x,y]\ \ldots$
- Test-Data-Selection
  - Test Data: $[3,9] = \mathrm{PUT}\ [3,9]$

    $[1,6] = \mathrm{PUT}\ [6,1]$
- Test-Driver Generation
  - SML driver
- Test Execution, Test-Documentation

Midi Example: Red Black Trees

Red-Black-Trees: Test Specification

```
testspec :
(redinv t  ∧
 blackinv t)

 →

    (redinv (delete x t)  ∧
     blackinv (delete x t))
```

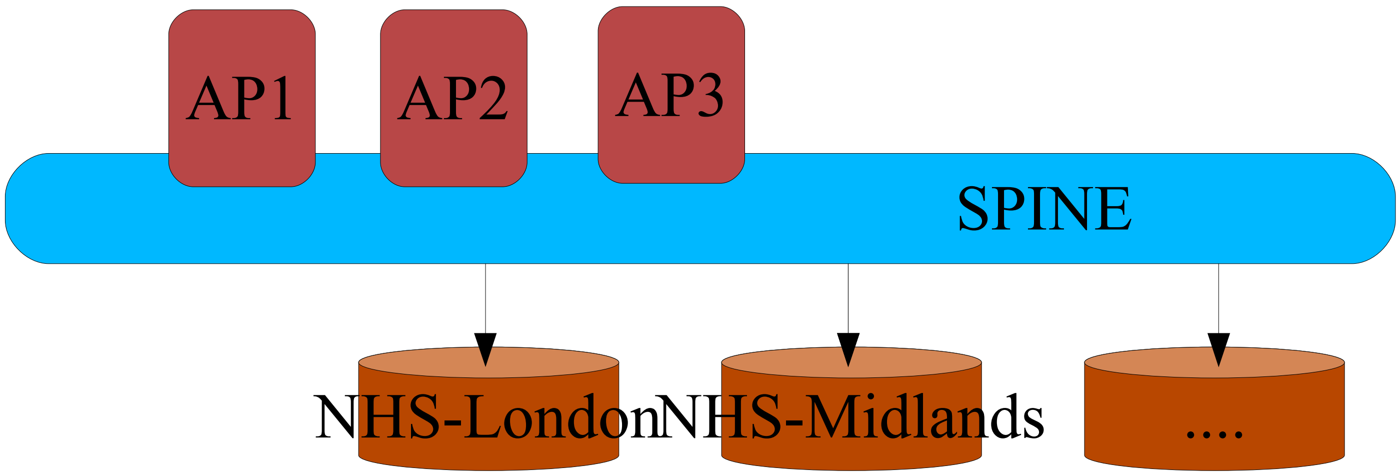
where `delete` is the program under test.

Large Example: Firewalls

- Modelisation
  - TCP-IP, nets and subnets, (stateful) firewalls, policies
- Test-Case-Generation from Test-Specification
  - Test-Sepcification: policy pkt = FUT pkt
  - Test-Cases: ... subnet x y $\Longrightarrow$ accept(http,x,d,y) = PUT(http,x,d,y)
- Test-Data-Selection
  - Test Data: accept(http,(132,17,24,12), "blob",(132,17,0,0))
- Test-Driver Generation
  - test-data fed into external driver [Diana Krueger 05)
- Test Execution, Test-Documentation
  - partially contained in our distribution

Case-Study: NPfIT

- Large Case-Study together with Britisch Telecom
- Test-Goal: NHS paptient record access control mechanism
- Large Distributed, Heterogenious System
- Legally required Access Control Policy (practically not really enforced)



Case-Study: NPfIT

- Modelisation
  - RBAC policies, Legitimate Consent, ...
- Test-Case-Generation from Test-Specification
  - Test-Specification: policy (AP1, sc, pat, op) = SPINE ...
  - Test-Cases: ... legitimite() $\Longrightarrow$ accept(AP1, sc, pat, op) = SPINE
- Test-Data-Selection
  - Test Data: . . .
- Test-Driver Generation
  - ?
- Test Execution, Test-Documentation
  - IPR

Case-Study: VAMP Processor

- Modelisation
  - registers, physical memory, processor-step-relation
- Test-Case-Generation from Test-Specification
  - …

- Test-Data-Selection
  - …

- Test-Driver Generation
  - automatic
- Test Execution, Test-Documentation
  - none

Code-Verification / Verifying Compilers

- Basis: Hoare Calculus + Dijkstra's wp calculus
- Specification in form of pre-post-condition programming language imperative
- Adaptions to realistic PL necessary (Java, C#, C (vanilla, X86 -o3, concurrent, ...)
- Can VERIFY a program wrt. spec for all input and all possible output !
- Needs massive automated theorem proving technology (Simplify, AltErgo, Z3, ...)

Code-Verification / Verifying Compilers

- Example:

$$\dfrac{true \to I \quad \dfrac{\dfrac{I \wedge x < 2 \to I'' \quad \overline{\vdash \{I''\}\ x :== x+1\ \{I'\}} \quad I' \to I}{\vdash \{I \wedge x < 2\}\ x :== x+1\ \{I\}}}{\vdash \{I\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{I \wedge \neg(x<2)\}} \quad I \wedge \neg(x<2) \to 2 \le x}{\vdash \{true\}\ \text{WHILE}\ x < 2\ \text{DO}\ x :== x+1\ \{2 \le x\}}$$

where $I'' = I'[x \mapsto x+1]$ and where we need solutions for

$A = true \to I$

$B = I \wedge \neg(x < 2) \to 2 \le x$

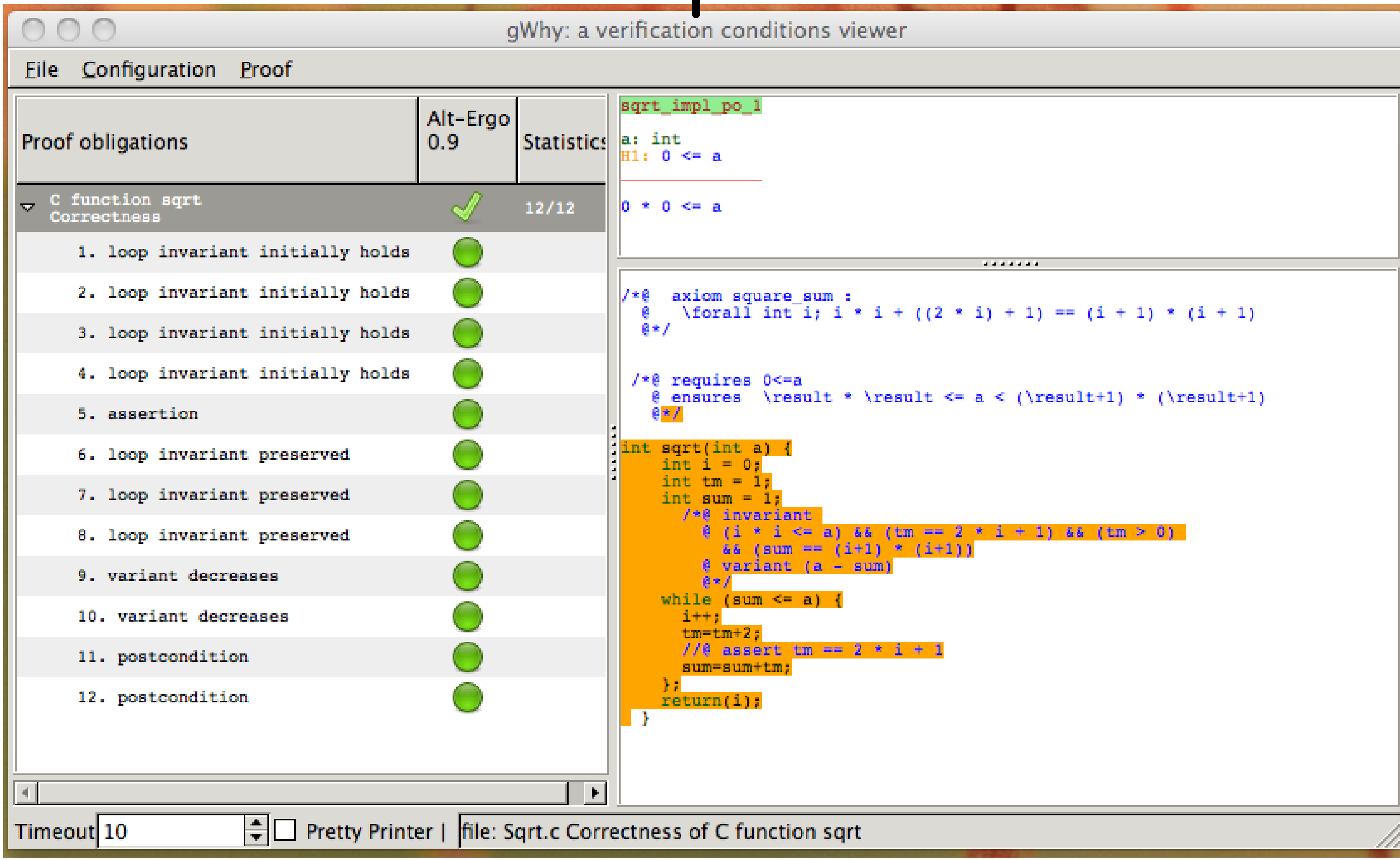
$C = I \wedge x < 2 \to I''$

$D = I' \to I$

Code-Verification / Verifying Compilers

- Microsoft Visual-Studio + Spec# + Boogie + Z3
  (for a C# like language)
- Microsoft Visual-Studio + VCC + Boogie + Z3
  (for a realistic subset of C / X86)
- gwhy + Why + AltErgo
- Eclipse + Jessy + Why + Z3 / AltErgo
  (Vanilla C)
- Isabelle/HOL + Simpl + …
  (Has a Vanilla C frontend)

Code-Verification / Verifying Compilers



Relevance of Formal Methods in Industrial Applications Today

- MDE

- MBT

- Code Verification by Automated Proof

Industrial Applications MDE

- The second-largest Software-Company

  SAP

  is in fact very MDE:

  - Business-Models of Companies were modeled in UML
  - own tool-chains generate data-base configs, tool-chains and entire web-services from that

  - little code is written by hand …

Industrial Applications MBT

- Windows 98-Server Protocol: the story so far

- 2000 : EU and US administration ruled Microsoft is a Monopoly in the Server Market (applying older Antitrust rules in the Telecommunication market)

- 2002 : EU required the "specification" of the server protocols in order to allow third-party vendors acces to the market

- Polished internal documents of Mocrosoft were considered "insufficient" by the EU referees …

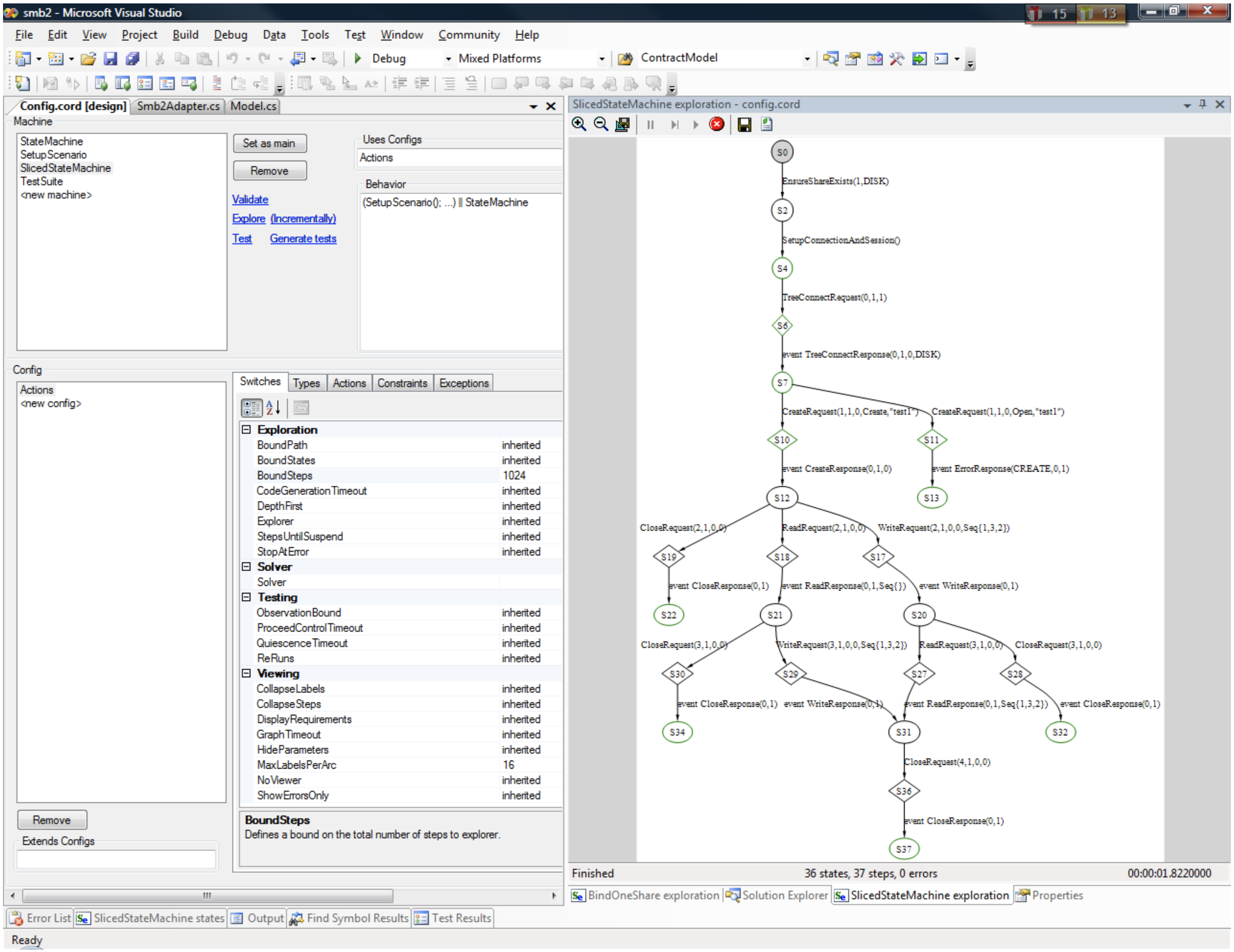
Industrial Applications MBT

- 2003: Microsoft legally contested this ruling, considering protocols as protected being IPR

- 2005: Microsoft lost the legal battle, was fined by 700 mio €, and forced to produce a document which:
  - also provides a formal specification
  - provides evidence that the model is actually compliant to the implemented system.

  Since then, a team of 200 people started to reverse engineer the Protocol (developed in 1995), essentially using a tool-family on the basis of Spec-Explorer
  … by the way, the team was located in Bangalore …

Industrial Applications



TestGen vs. Spec-Explorer

- HOL-TestGen offers a similar approach
  to SE process integration (albeit on a smaller scale ...)

- Unlike e.g. Spec-Explorer (by Microsoft, available as VisualStudio Plugin), it emphasizes (well, we are academic ;-)):

  - logical cleaness and an expressivness.
    Modeling Language HOL instead of, say, an OO-language with quantifiers

  - symbolic computations having their roots in Theorem Proving instead of plain enumeration and model-checking

Industrial Applications – MBT

- Windows Server 98 Protocol :

  Wolfgang Grieskamp[08]:

  Using Model-Based Testing for Quality Assurance of Protocol Documentation

  Invited Talk MBT 2008, Budapest.
  http://research.microsoft.com/users/wrwg/MBTETAPS.pdf

Industrial Applications – Code Verification by Proof

- Hardware Suppliers:
  - INTEL: Proof of Floating Point Computation compliance to IEEE754 (Forte-System)
  - INTEL: Correctness of Cash-Memory-Coherence Protocols
  - AMD: Correctness of Floating-Point-Units againt Design-Spec (ACL2)
  - GemPlus: Verification of Smart-Card-Applications in Security (Coq)

Industrial Applications – Code Verification by Proof

- Software Suppliers:
  - Microsoft: SAL Annotations (a limited form of pre-postconds restricted to memory properties) has been used to specify the entire Vista/Windows7 Code-Base (… and MS Office, too). 15 MLocs Code !!!
  - MicroSoft: Many Drivers running in „Kernel Mode" were verified
  - MicroSoft: Verification of the Hyper-V OS (60000 Lines of Concurrent, Low-Level C Code …)
  - NICTA: L4-Verified Project Verified a Mach Kernel
  - Pike-OS Verification

…

Relevance for Emmerging Countries ?

- No Modern Hardware without Verification Techniques (SAT, BDD, HOL, ACL2)
- Software Specifications will turn up in Outsourcing Scenarios
- Model-based Testing IS ALREADY APPLIED IN INDIA ...

A Perspective for Teaching at ICT - IIT Rajastan

- Teaching Proving (Interactive & Automated) is a Prerequisite for Scientific Engineering (Phd's should have learned it, even if they don't do it professionally)
- Teaching Tool-oriented Verification
  - for Hardware
  - for protocols in services
- Teaching Model-based Testing for a controlled, quality-oriented Software-Development Process

Conclusion

- Formal Methods ARE relevant for Emmerging Countries !!!
  - Model-based Testing (see next)
  - Interactive Proof Techniques for Teaching (see next)
  - Automated Theoremproving is highly relevant for Hardware-verification
  - Automated Theoremproving is relevant for (high-quality) Software-verification

Conclusion

- Model-based Testing allows:
  - development of Modeling Capabilities fundamental for Advanced Software Engineering
  - Key-Technique for Globalized Software Production !
  - Expertise in automated Testing Soft- and Hardware, even in presence of heterogeneous or legacy code

Conclusion

- Protocol Analysis allows:
  - establishing deadlock-freeness or
  - … security properties in protocols
  - … and protocol implementations

Conclusion

- The ITP Programme (and Isabelle in particular, which I consider a leading edge) allows:
  - reconciliation of foundational with pragmatic technology issues
  - reconciliation specification & programming
  - proved feasibility of proof architectures of considerable size

Conclusion

- Reusing Isabelle as FM tool foundation offers:
  - substantial conservative libraries
  - standardized interfaces to tactic and automatic proof
  - proof documentation
  - code generation
  - a programming interface and genericity in design

  … a lot of machinery not worth to reinvent.