



Model Driven Engineering Technologies

# Adaptation of Model Based Testing to Industry

Alan Hartman  
IBM Research - Haifa Labs



IBM Haifa Labs

Agile and Automated Testing Seminar  
Tampere University of Technology  
August 15 2006



## Acknowledgements

---

- The Tampere University of Technology
  
- My colleagues:
  - ▶ Kenneth Nagin
  - ▶ Sergey Olvovsky
  - ▶ Andrei Kirshin
  - ▶ Leonid Raskin
  - ▶ Mila Keren
  
  - ▶ Mika Katara
  - ▶ Clay Williams, Avik Sinha, Amit Paradkar
  - ▶ Mark Utting and Bruno Legeard
  
  - ▶ Jeff Schuster, Glenn Hughes



## Agenda

---

- Acknowledgements
- What is Model Based Testing (MBT)?
- Why and why not do MBT?
- How is MBT done?
  - ▶ Different styles of Model Based Testing
- MBT in Industry
  - ▶ Report on MBT efforts over the years
  - ▶ Success criterion for technology innovation
  - ▶ Where are we (IBM Rational and IBM Research) going?
- Conclusions
- Q & A or Free Discussion (feel free to interrupt!)



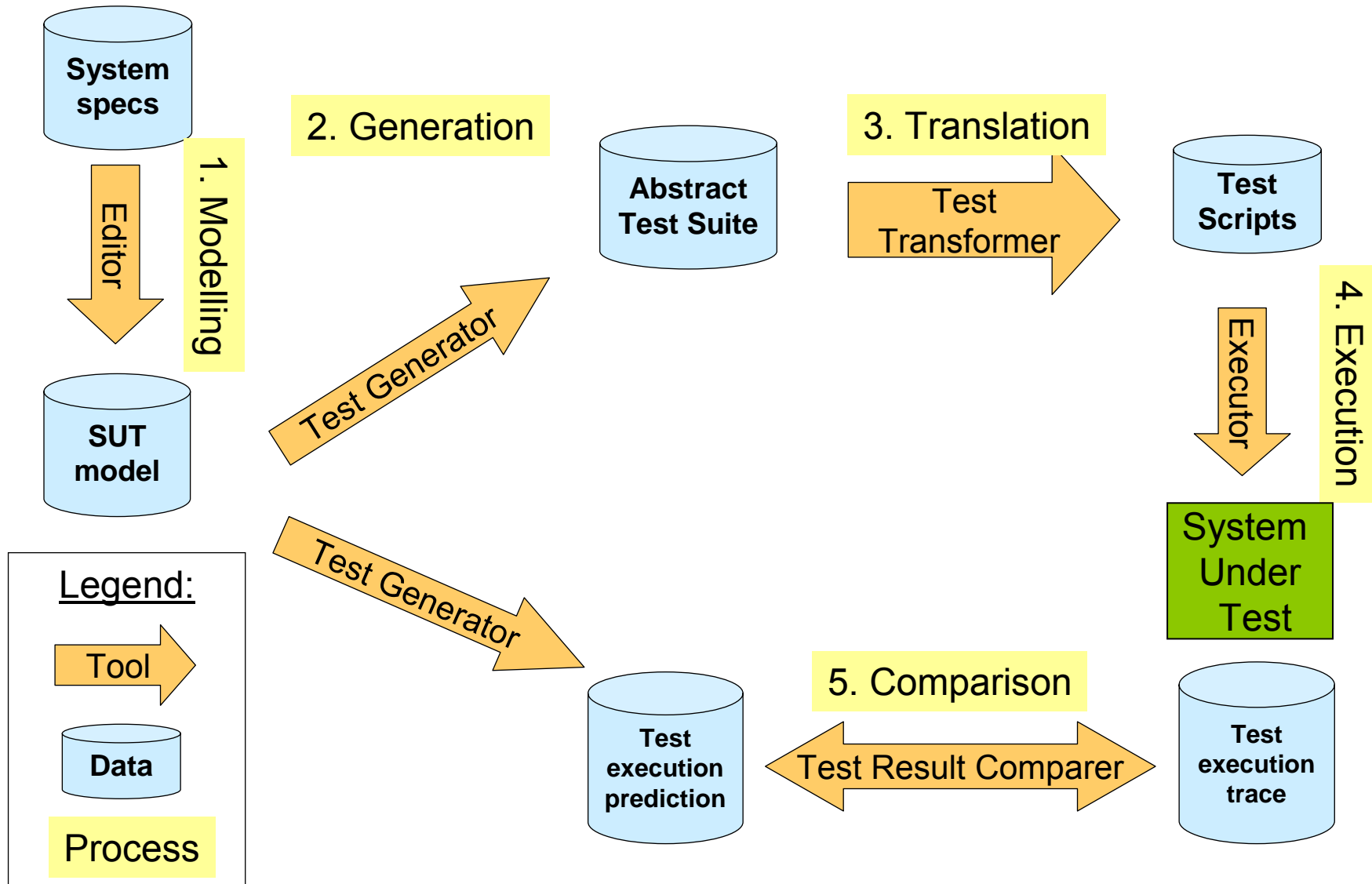
## Model Based Testing – A Definition

---

- The easy answer:
  - ▶ All testing is based on a model
  
- **Model Based Testing** occurs when the model is:
  - ▶ formalized
  - ▶ recorded in some form
  - ▶ used for generating test cases or oracles
  
- **Model Driven Testing** is a special case of MBT
  - ▶ Ideas rooted in the OMG's Model Driven Architecture (MDA) initiative
  - ▶ **Abstract** models – e.g. platform independent, high level, black box, etc.
  - ▶ **Automated** test transformations



# What is MBT? (The MBT Methodology)

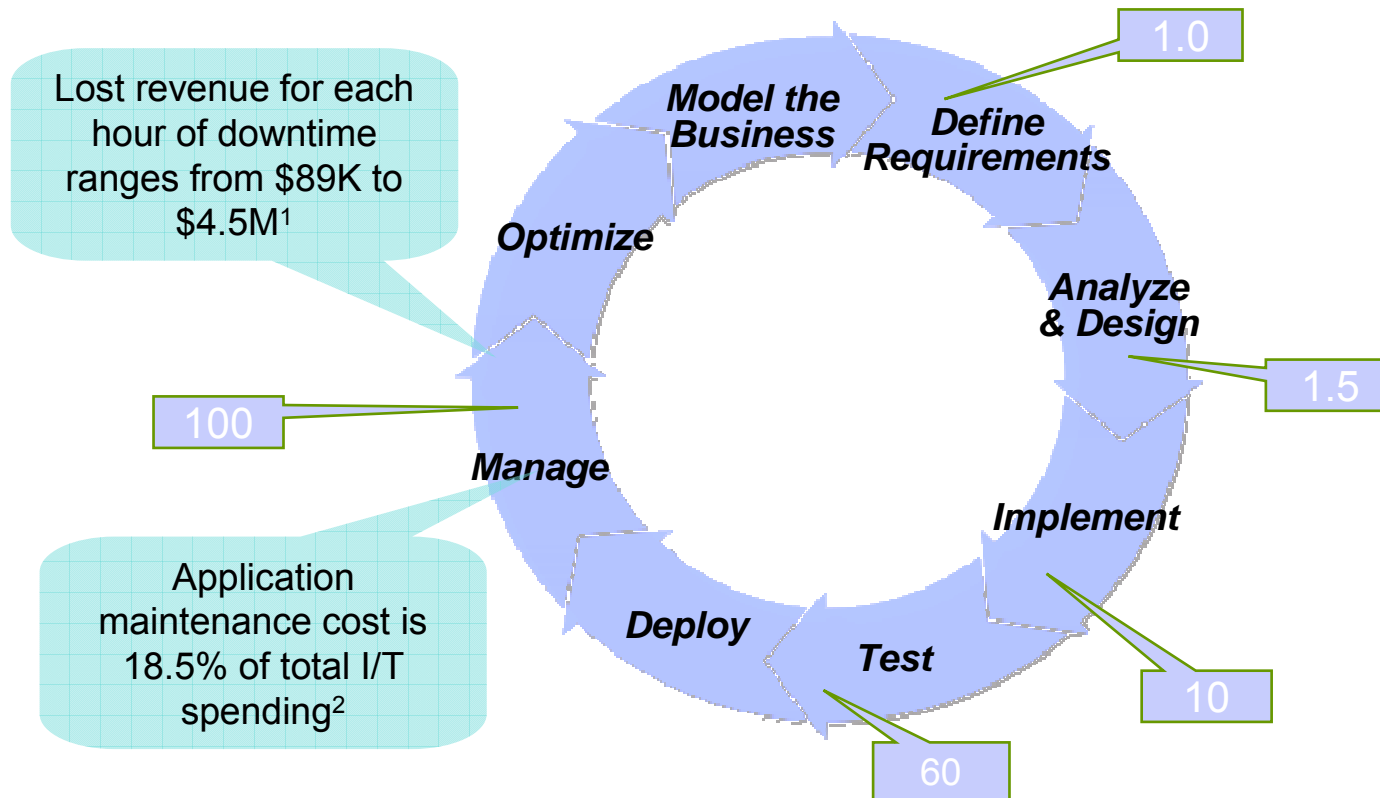




## Why do Model Based Testing?

### **Late defect discovery exponentially increases repair costs**

- An IBM & Gartner “rule of thumb” for the relative costs to fix defects

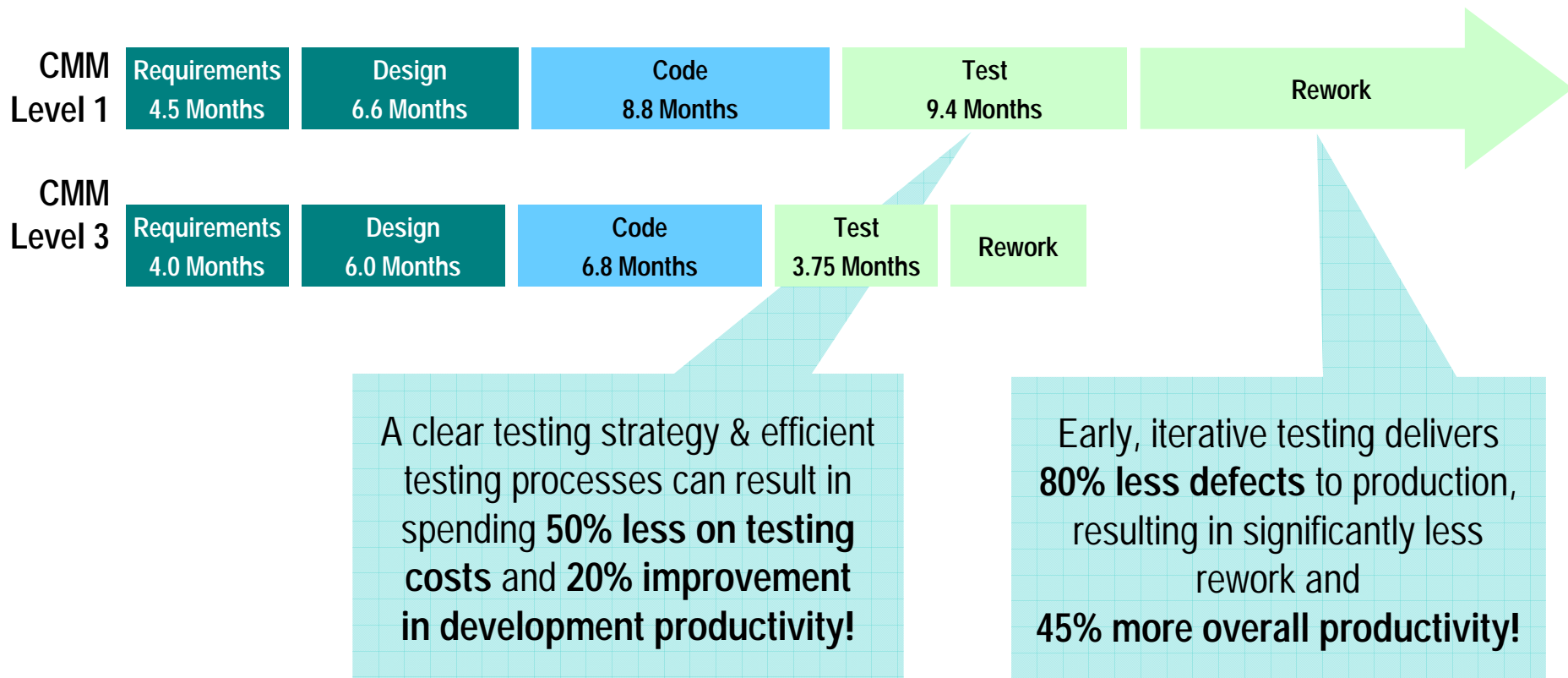


<sup>1</sup> AC Market Analysis: Self-healing market June 2004

<sup>2</sup> Source: Tivoli Marketplace Assessment, February 2004



## Mature processes & integrated tools



Note: CMM Level 1 organizations spend 30%+ of project dollars on testing.  
CMM Level 3 organizations spend 15%.



## Why do Model Based Testing?

---

- **It's not only a sales pitch – these are real advantages!**
- **Starting from specification**
  - ▶ Shortens the development process
  - ▶ Teams testers with developers
  - ▶ Forces testability into product design
- **Building behavioural model and test interface**
  - ▶ Finds design and specification bugs - before the code exists
  - ▶ The model is the test plan - and is easily maintained
- **Automated test suite generation**
  - ▶ Coverage is guaranteed - increases testing thoroughness
  - ▶ Zero test suite maintenance costs
  - ▶ Automatic prediction of test results
- **Automated test suite execution**
  - ▶ Finds code and interface bugs
  - ▶ Reduces test execution costs



## Why not do MBT? - Industry Adoption Pains (1)

---

- **Methodology Complexity**
  - ▶ Practitioners need to:
    - Model
    - Understand Test Generator limitations
    - Code (to translate abstract test cases)
  - ▶ These are skills of senior developers, not average testers
  
- **Test Generation Limitations – real and imagined**
  - ▶ State explosion – small models or weak coverage?
  - ▶ Loss of control
  - ▶ Test generation tool imposing limitations on modeling language



## Why not do MBT? - Industry Adoption Pains (2)

---

- **Weak Usability of Tools**

- ▶ Model debuggers?
- ▶ Model libraries?
- ▶ Testing of models?

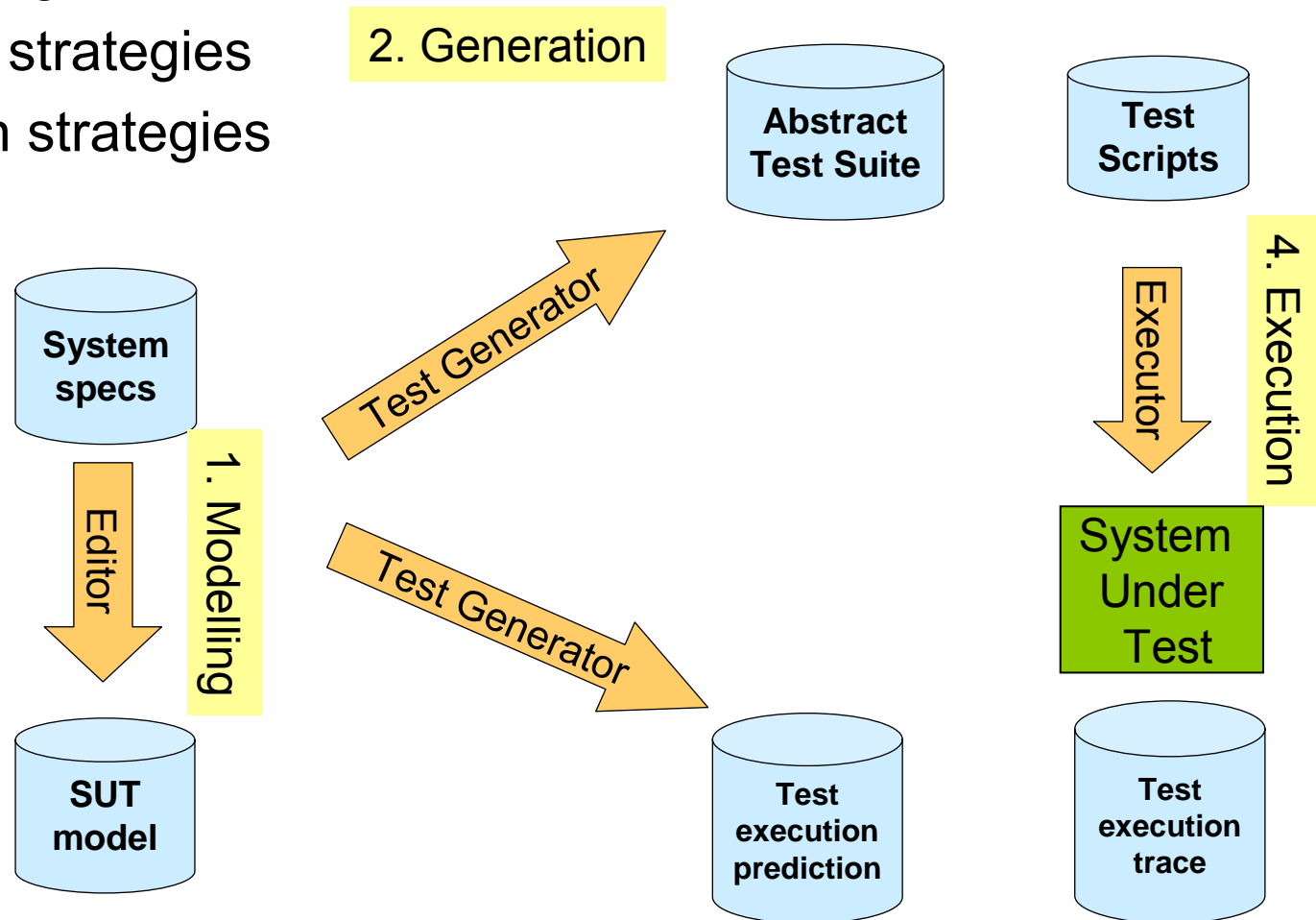
- **Organizational Challenges**

- ▶ Revolutionary: total process reorganization
  - Early availability of testers
  - More contacts with specification writers and architects
- ▶ Low level rejection:
  - Testers face work redefinition
  - Old (often locally developed) tools are discarded
- ▶ You need two champions: high level manager & expert tester



## How is MBT done?

- Modeling strategies
- Test selection strategies
- Test execution strategies

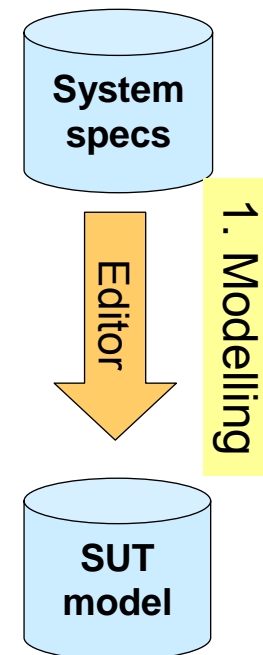




## Modeling Strategies (Utting and Leguard Classification)

---

- Pre and post condition models (State based, data flow)
- Transition based models (FSM)
- Trace based models (MSC, LSC)
- Algebraic models (Function based)
- Stochastic models (User profile based)





## Modeling Strategy 1 – Pre and Post Conditions

---

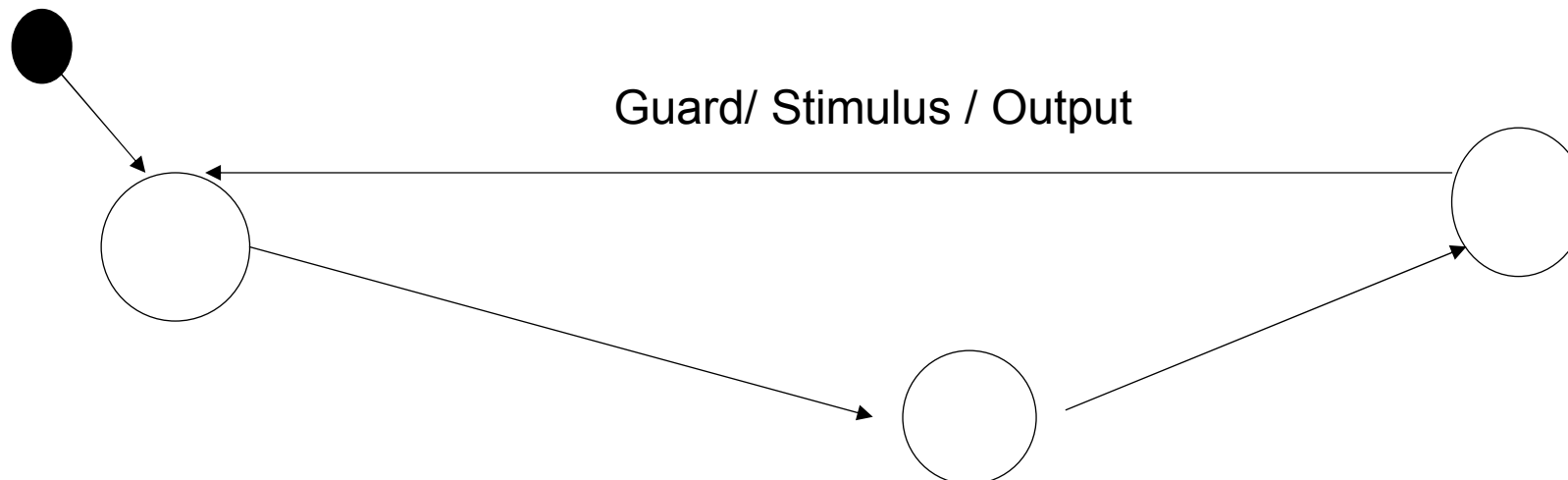
- Examples: Spec#, JML, B, UML with OCL, Z
- Main modeling decisions:
  - ▶ Choice of state variables
  - ▶ Choice of stimulus granularity
- Often used for data intensive systems
- Typically text based



## Modeling Strategy 2: Transition based models

---

- Typically graphical:
- Tabular notations are also used
- Finite State Machines
- Labelled transition systems & IOLTS
- Statemate/Simulink
- UML State machines - Statecharts
- Often used for reactive systems and UI based testing





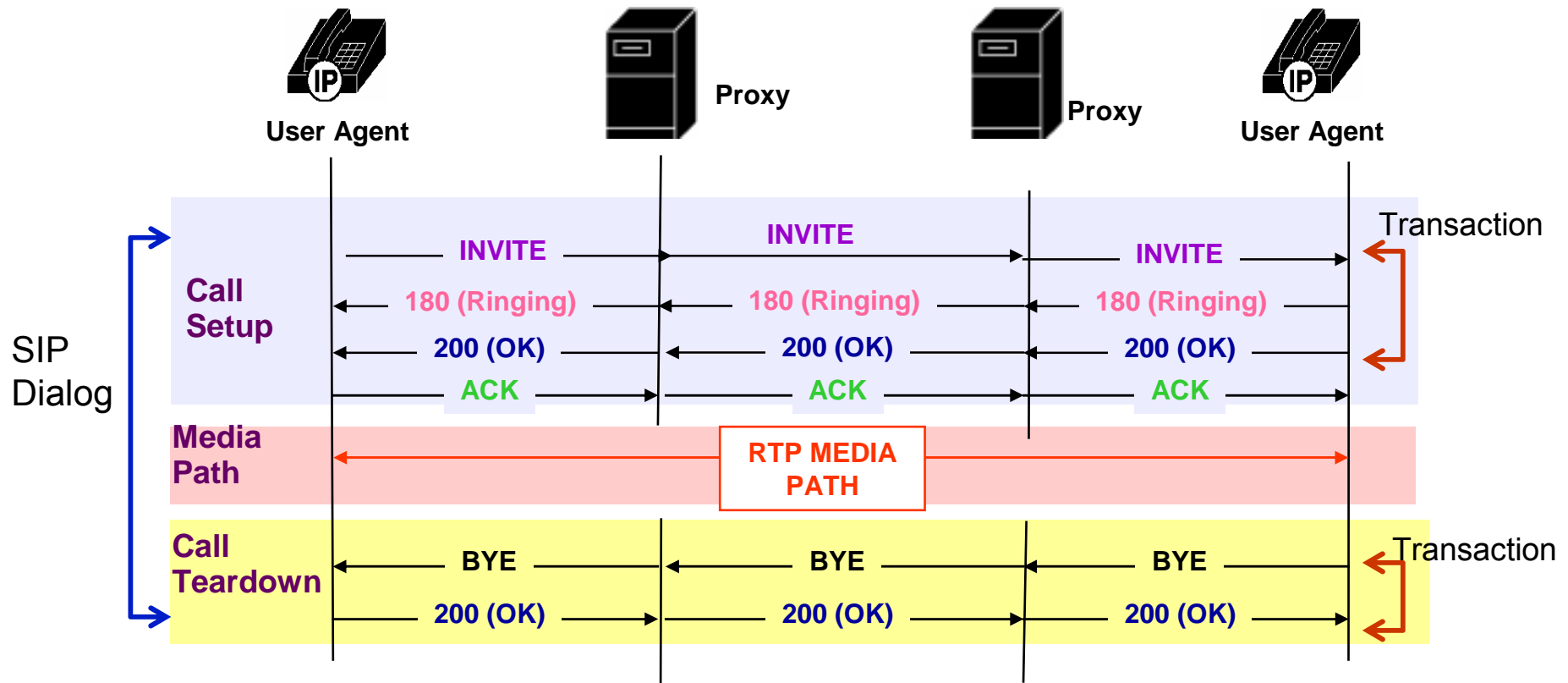
## Modeling Strategy 3: Trace based models

---

- SDL
- Message sequence charts
- UML interaction diagrams
- Live sequence charts (MSCs with pre and post conditions and anti sequence charts)
  
- Describe the allowable traces of the systems
- Cannot describe all behavior – although the play engine attempts to do so
  
- Better for describing test cases

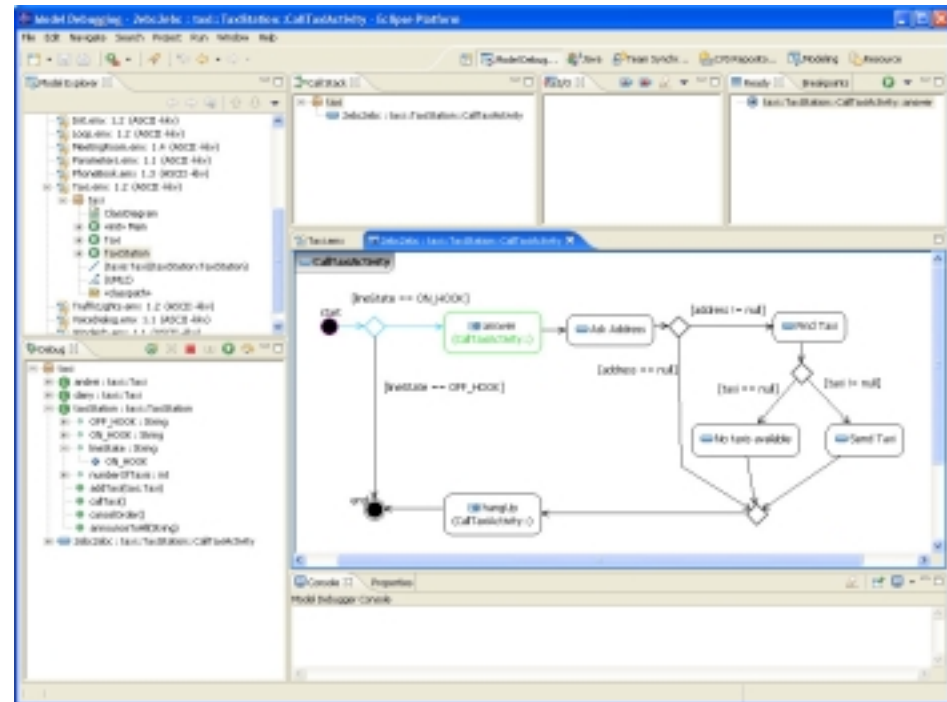


# Trace based notations - example



## Modeling strategy 4: Function / Algebraic /Operational Models

- Uses first order (or higher order logic) to describe behavior
- CSP
- HOTTest
- Process algebras
- Petri nets
- Text based (except Petri nets)





## Modeling strategy 5: Statistical / Combinatorial modeling

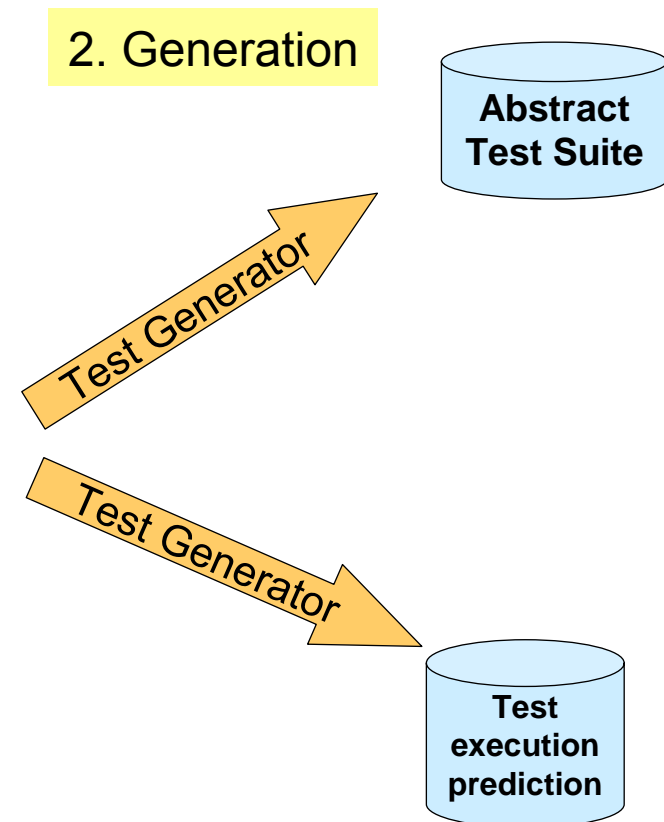
---

- Describing user profiles, input sampling
- Describes input – not behavior
  
- Clean room methodologies
- JUMBL
  
- Pairwise and higher
- AETG
- Etc. etc. etc.

## Test Selection Strategies

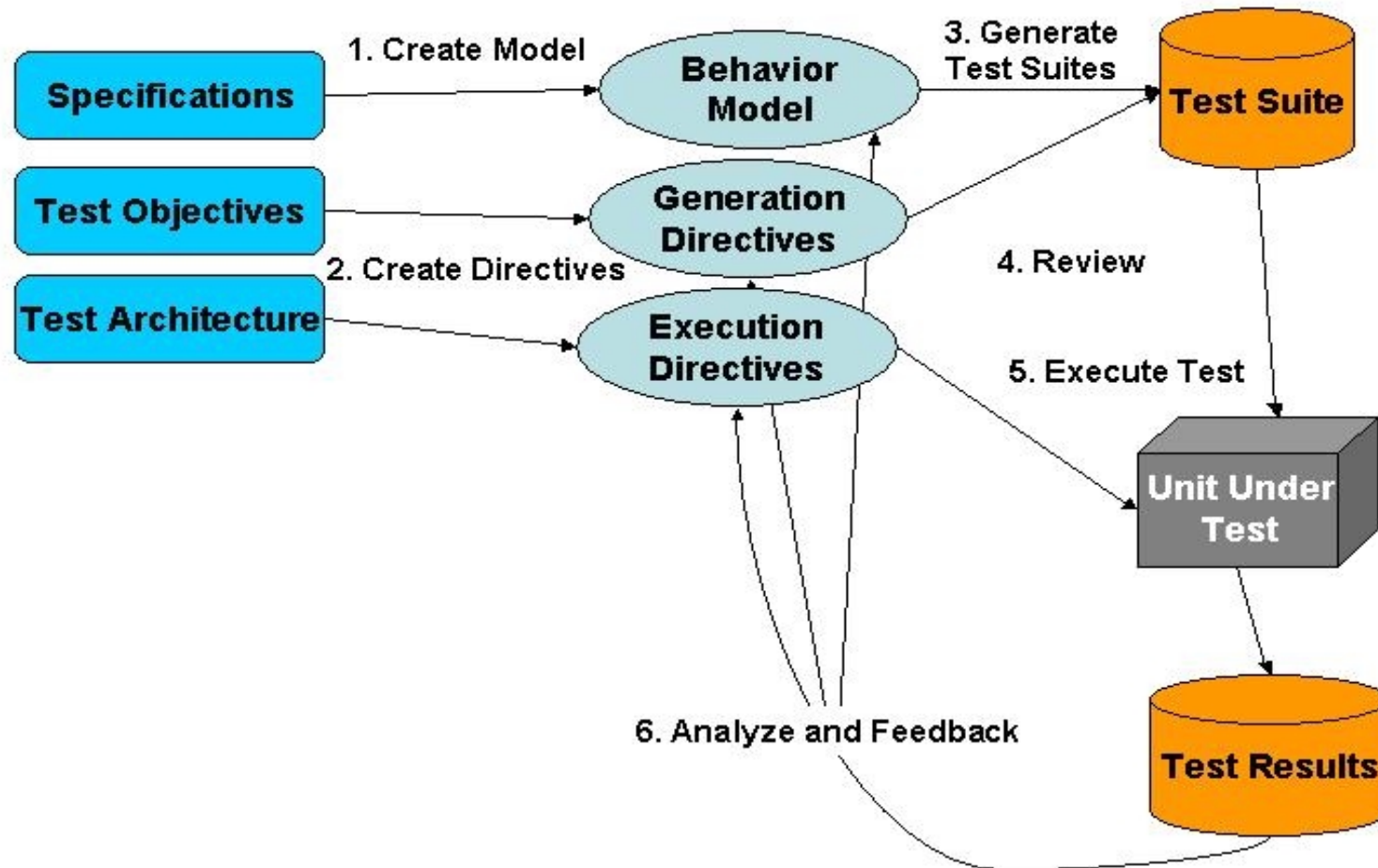
---

- Mostly depend on the modeling strategy – the aim is to cover the model
- Imitate traditional coverage strategies
  - ▶ Pre and Post → Condition coverage
  - ▶ Transition Based → State coverage and transition coverage
  - ▶ Trace Based → Test purposes
  - ▶ Algebraic → Operator coverage
  - ▶ Statistical → Frequent behavior coverage
  - ▶ Combinatorial → Input coverage



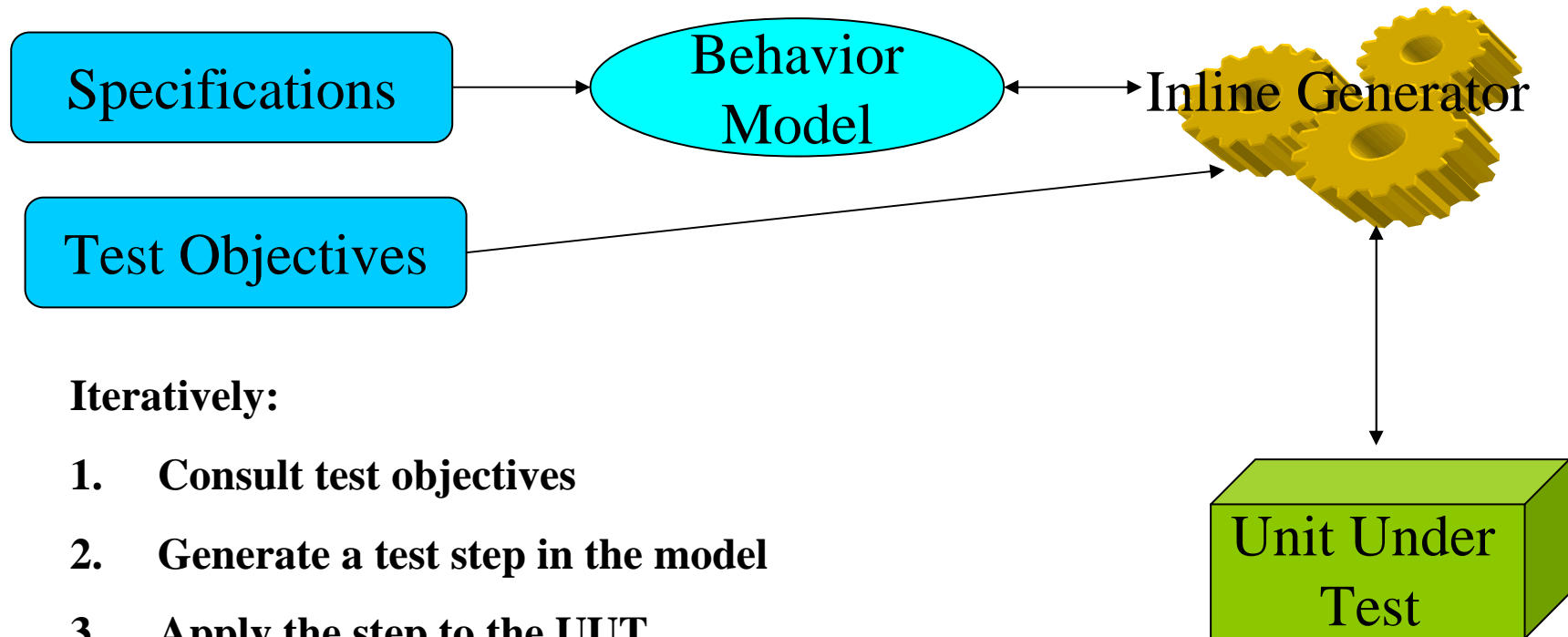


## MBT Execution Strategy 1 - Offline Model Based Testing





## MBT Execution Strategy 2 - Online Model Based Testing



### Iteratively:

1. Consult test objectives
2. Generate a test step in the model
3. Apply the step to the UUT
4. Observe the result
5. Consult the model for validation



## Research results on model based testing

---

- The number of combinations of modelling, selection and execution strategies is large (5X5X2)
  
- Whatever variation you choose:
  - ▶ Someone somewhere has found an example where that method works better than all others
  
- Conferences and journals tend to publish success stories
  
- Keynote speakers have more freedom



## Personal Experiences

---

- 1997 – GOTCHA for processor architecture verification
- 1998 – IBM US PoC with very experienced tester
- 1999 – IBM US file system test
- 2002 – AGEDIS project, FT, DB application, Messaging system
- 2001 – IBM Telephone company engagement
- 2003 – Health care ISV
- 2005-6 – Custom made solutions tailored to industry



## What is the reality? Part 1

---

- 1997 – GOTCHA for processor architecture verification
- Coverage driven testing
- Tools were used by Ph. D.s
  - ▶ “Successful” in the lab., not production
  - ▶ Hardware not software



## What is the reality? Part 2

---

- 1998 – IBM US Proof of Concept with **very experienced** tester
- First funding for our under-cover project in SW testing
- Reality hits
  - ▶ Violent resistance
  - ▶ I can do better by hand
  - ▶ Bad UI



# 1999 – IBM US file system test (Reality Part 3)

- Retest of functions
- Modelling and translation by testers
- Comparison
  - ▶ Original test: 18 bugs, 12 PM
  - ▶ Pilot test: 15 original bugs + 2 escapes, 10 PM (INCLUDING learning curve)
- Conclusions:
  - ▶ Efficient way to free the tester for creative testing
  - ▶ Replaces a large part of the manual test case writing
- Reality:
  - ▶ Never used again

DEFECTS BY SEVERITY	#	%
1	0	0
<b>2</b>	<b>10</b>	<b>58.8</b>
3	6	35.2
4	1	5.8

DEFECTS BY ODC TRIGGERS	#	%
Coverage	6	35.2
Variation	1	5.8
<b>Sequencing</b>	<b>8</b>	<b>47.0</b>
Interaction	1	5.8
Load	1	5.8



## Was the problem our modelling language?

---

- GDL was clunky, looked like Pascal, came from hardware, not sexy
- 2002 – AGEDIS project - move to UML
- Reality Part 4
- Tool collaboration
  - ▶ Produced a camel rather than a racehorse
- Three industrial teams
  - ▶ Another Ph. D.
  - ▶ A below standard test engineer
  - ▶ A genius with the heart of a toolmaker (the NIH - Not Invented Here - problem)



## What is the reality? Part 5 – Beginning to see the light

---

- 2001 – IBM Telephone company engagement
- Rapid deployment
  - ▶ Team of five crack testers with a tight deadline
  - ▶ Enormous volume of testing
- Created an automated process for converting existing Cobol artifacts into GDL models
- **The reality**
  - ▶ Mountains of bugs uncovered
  - ▶ 60% of the bugs were documentation bugs
  - ▶ No repeated use of the tools



## What is the reality? Part 6 – Beginning to see the light

---

- 2003 – Health care ISV
- Conversion of existing testing artifacts into GDL models
- Simplification of the UI
- Modelling using a spreadsheet
- Simplified coverage criteria
- **The reality**
  - ▶ Our champion at the ISV got a promotion
  - ▶ No repeat business





## MBT Custom Solution

---

- Parsing scenario tables creates a Gotcha model
- Test generation creates sequences of scenarios
- WinRunner Scripts reused as is
- Benefits:
  - ▶ No manual selection
  - ▶ Coverage models used
  - ▶ MBT – totally hidden from end users



## The reality check

---

- MBT has been around for at least 7 years
- Current number of software efforts that use MBT:
  - ▶ No studies
  - ▶ Guess 1 - 5 %, closer to 1%.
- Industry interest exists – but adoption levels do not seem to grow significantly



## What is the reason?

---

- Tools problem?
- Service problem?
- Bad marketing?
  
- Methodology problem?
  - ▶ Any pure MBT solution regardless of the tools too hard?
- Wrong business model?



## Where is IBM Research going?

---

- Simplification:
  - ▶ Domain specific solutions
    - UML profiles
    - Ready made translations
  - ▶ Reuse of existing artifacts
    - IDL descriptions
    - Industry Standard protocols
- Business model:
  - ▶ MBT as a service
  - ▶ Creating custom made solutions for
    - Applications
    - Domains
- Integration with development environment



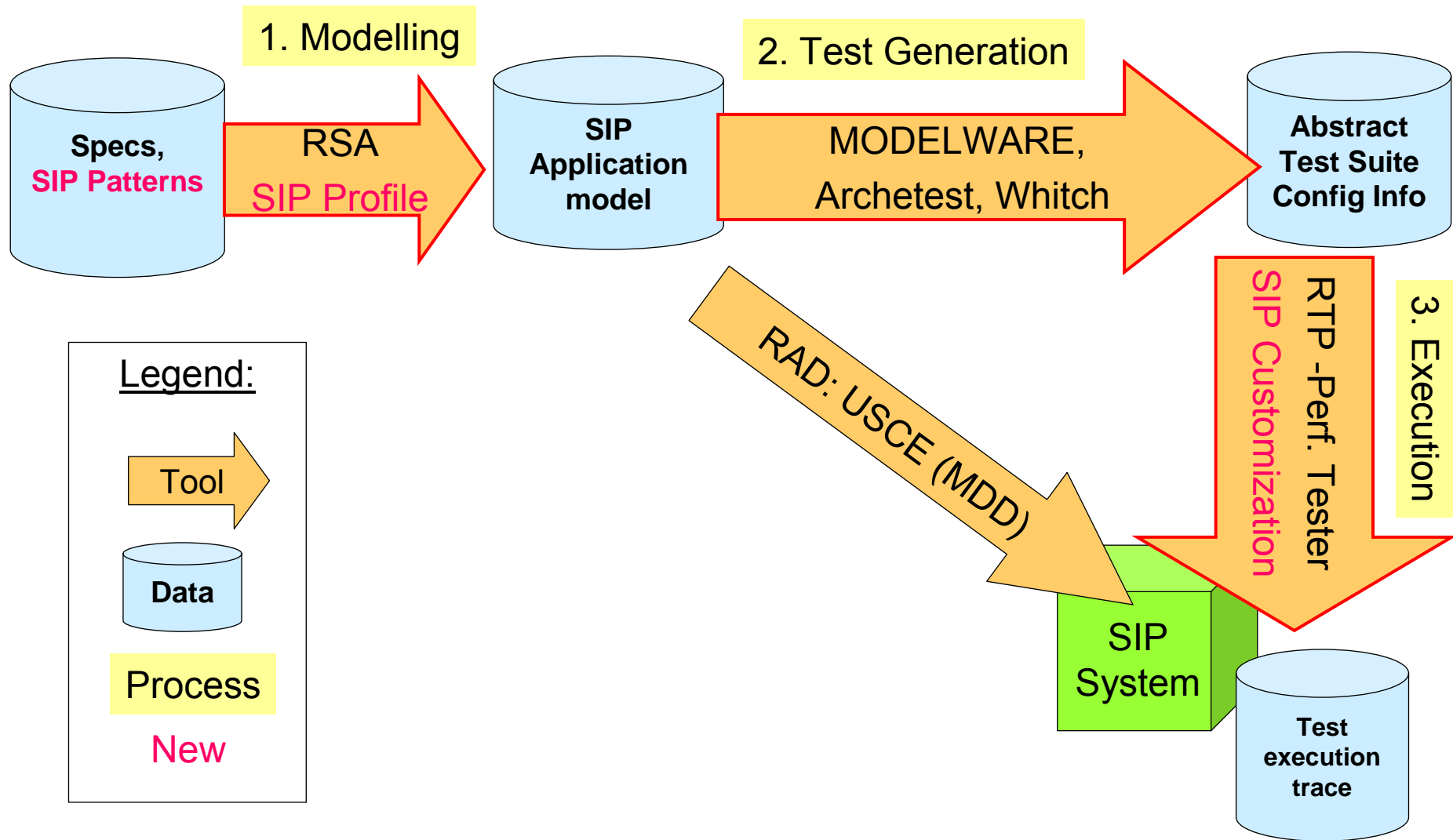
## SIP Application Testing Project – Solution Components

---

- **Test Execution**
  - ▶ SIP enablement of Rational Performance Tester
- **Test Creation**
  - ▶ Manual test specification
  - ▶ SIP enhancement of UML interaction diagrams
  - ▶ TTCN-3 import
  - ▶ UML profile for SIP
  - ▶ Test generation from SIP application models
- **Methodology**
  - ▶ RUP plugin
  
- **Part of lifecycle toolset for SIP applications**



# Model Based Testing for SIP - integrated with Rational Tools





## IBM Rational Focus: Business Driven Development

---

### *Software quality best practices*

#### Best Practices

**Focus on quality early**

**Integrate quality into the life cycle**

**Govern testing to ensure compliance**

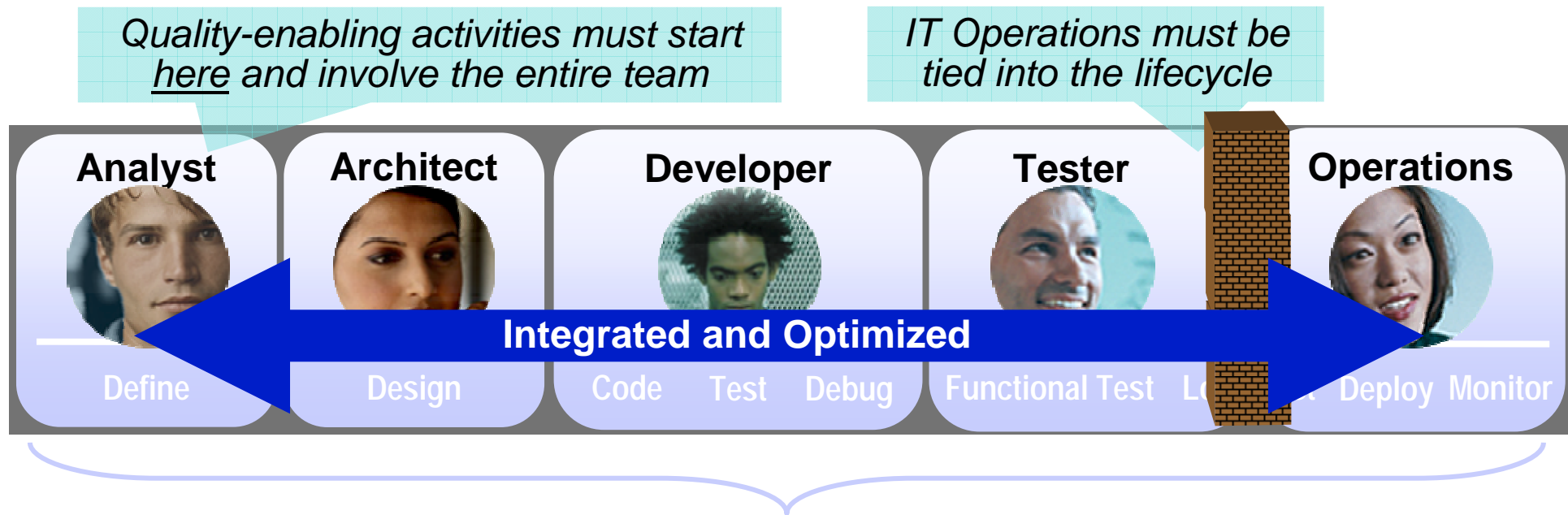
**Support testing of modular systems**

**Leverage open standards**



## Quality is a continuous, iterative, integrated process

***Prevent, detect, diagnose and remove defects***

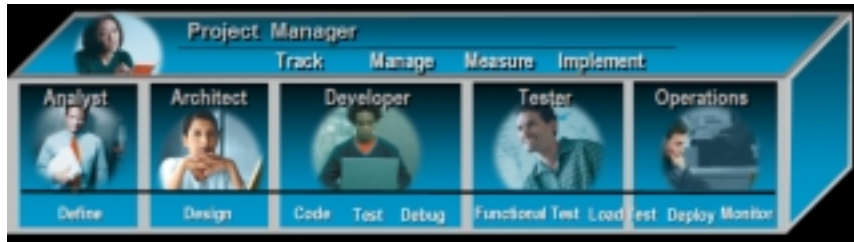


**More than defect repair –  
it is proactive, closed-loop software fitness process**



# Rational SW Quality Product Roadmap – Past and today

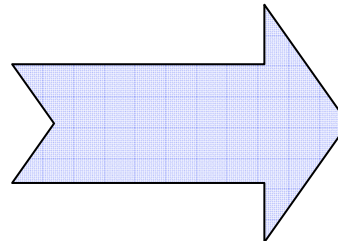
2004 - 2005



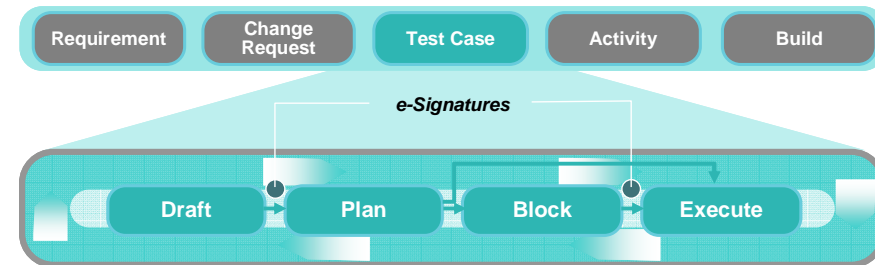
## Continuously Ensure Quality

### Value for Customers:

Common eclipse platform for practitioner tools and life-cycle integration with Requirements, version control and defect management tools



2006 - Today



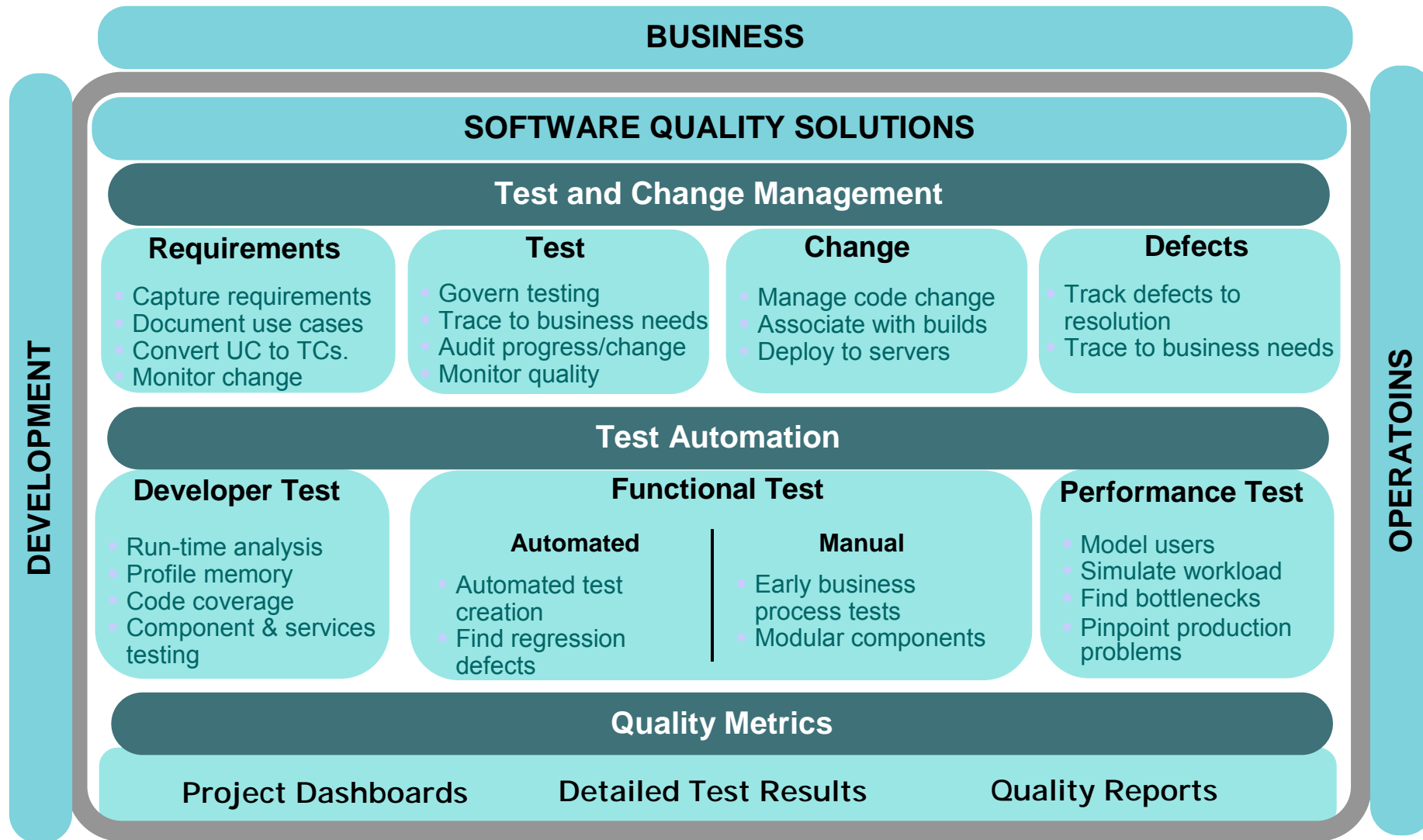
## Enterprise Test Management

### Value for Customers:

Extensible test ecosystem  
 Single project view  
 Global project coordination  
 Configurable, enforceable process

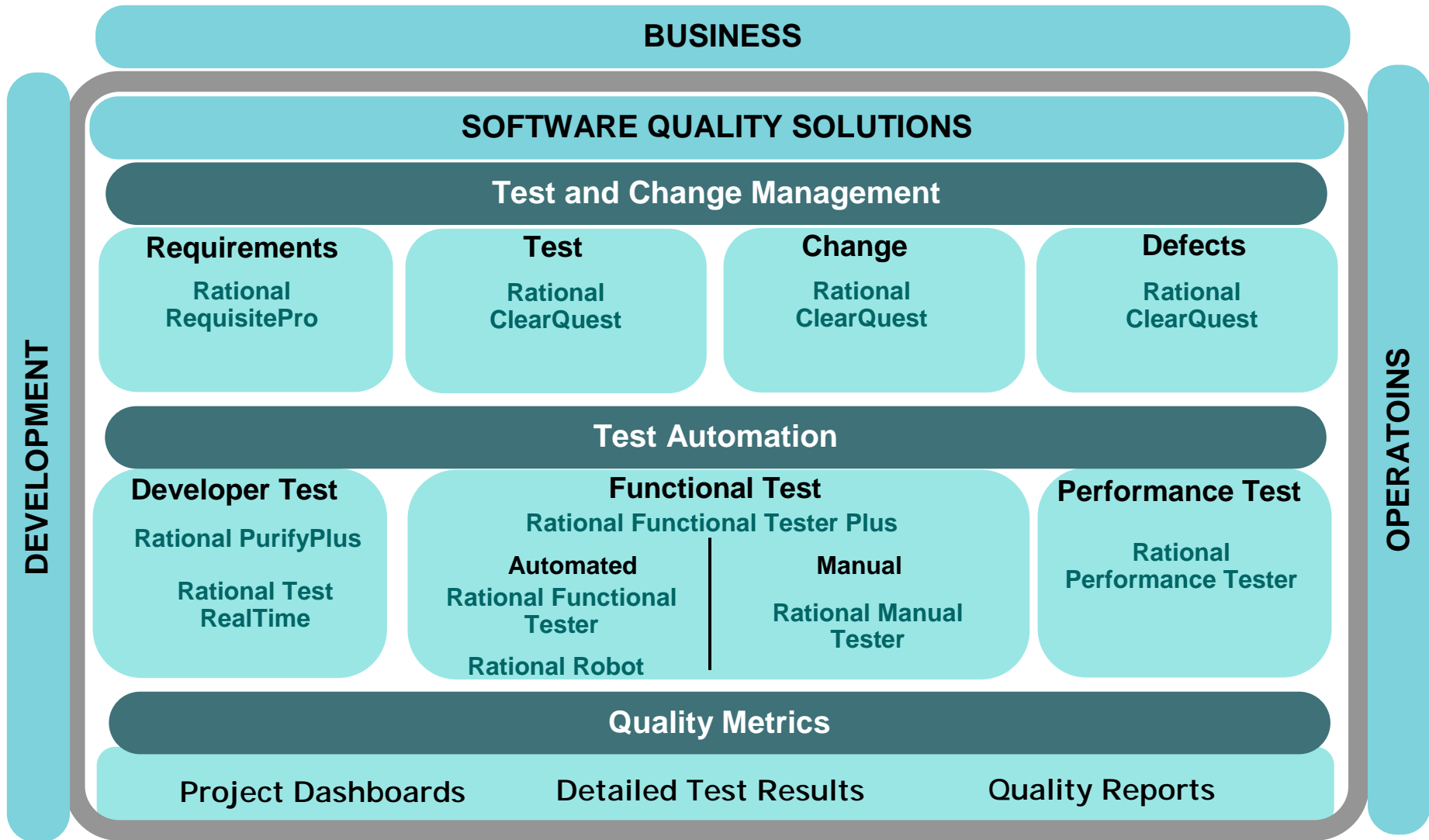


# Rational Software Quality Challenges





# Rational Software Quality Solutions





## Conclusions

---

- Testing is not the issue
- Methodology is not the issue
- Tools are not the issue
- **Software quality** and developer/tester **productivity** are the essence of business driven development
- **Modeling** is a proven method of dealing with complexity
  - ▶ Abstraction
  - ▶ Automation





Model Driven Engineering Technologies

Thank You.

Contact:

[hartman@il.ibm.com](mailto:hartman@il.ibm.com)

IBM Haifa Labs

Agile and Automated Testing Seminar  
Tampere University of Technology  
August 15 2006