

Test Generation towards TTCN-3

Paul Baker

Background

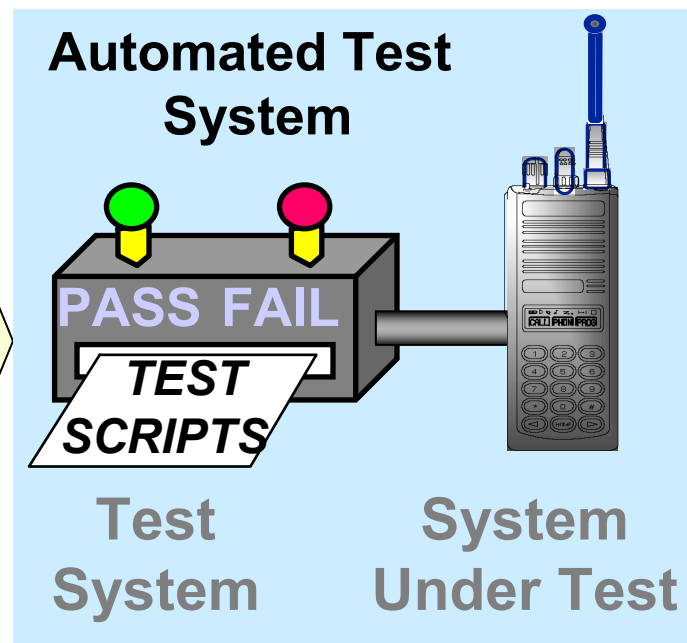
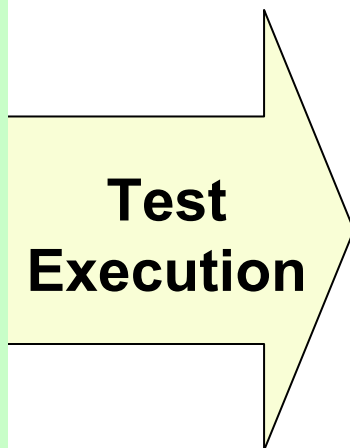
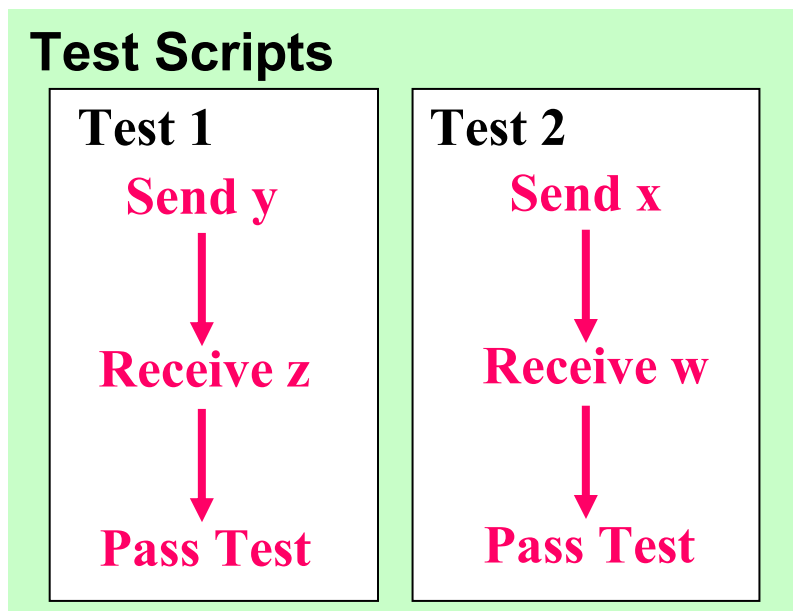
- **Model-Driven Development (MDD)**
 - Motorola's is a maturity MDD company
 - Mature software company:
 - SEI CMM & CMMI levels range between 3-5
- **Standards Participation**
 - *ITU-T/ETSI*: MSC, SDL, TTCN-3 (GFT),
 - *OMG*: UML 2.0 & UML 2.0 testing profile
- **Requirements-based Test Generation & Verification**
 - MSC2000 and UML 2.0 test generation and verification

Typical Test Process

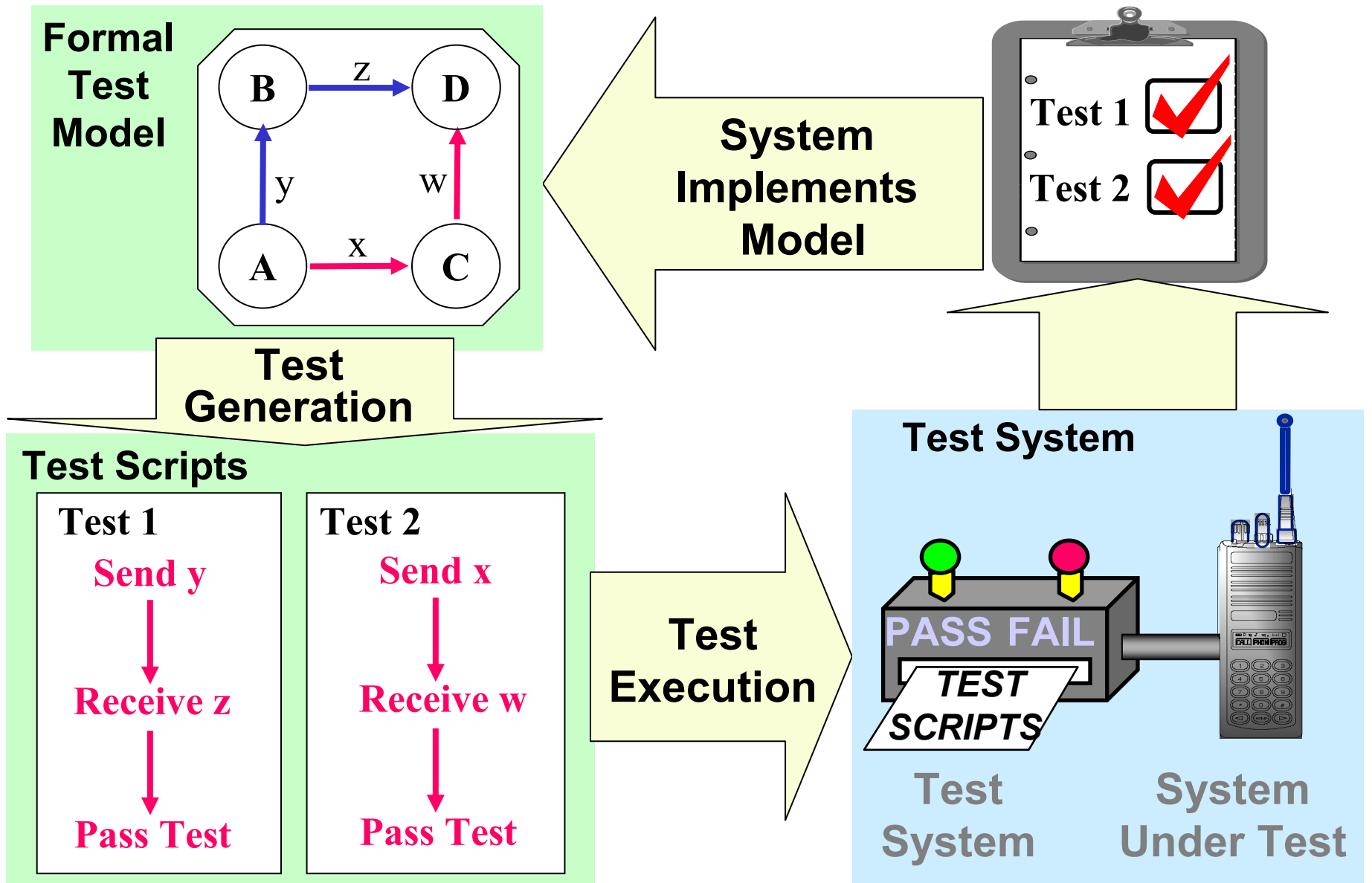


Test Development

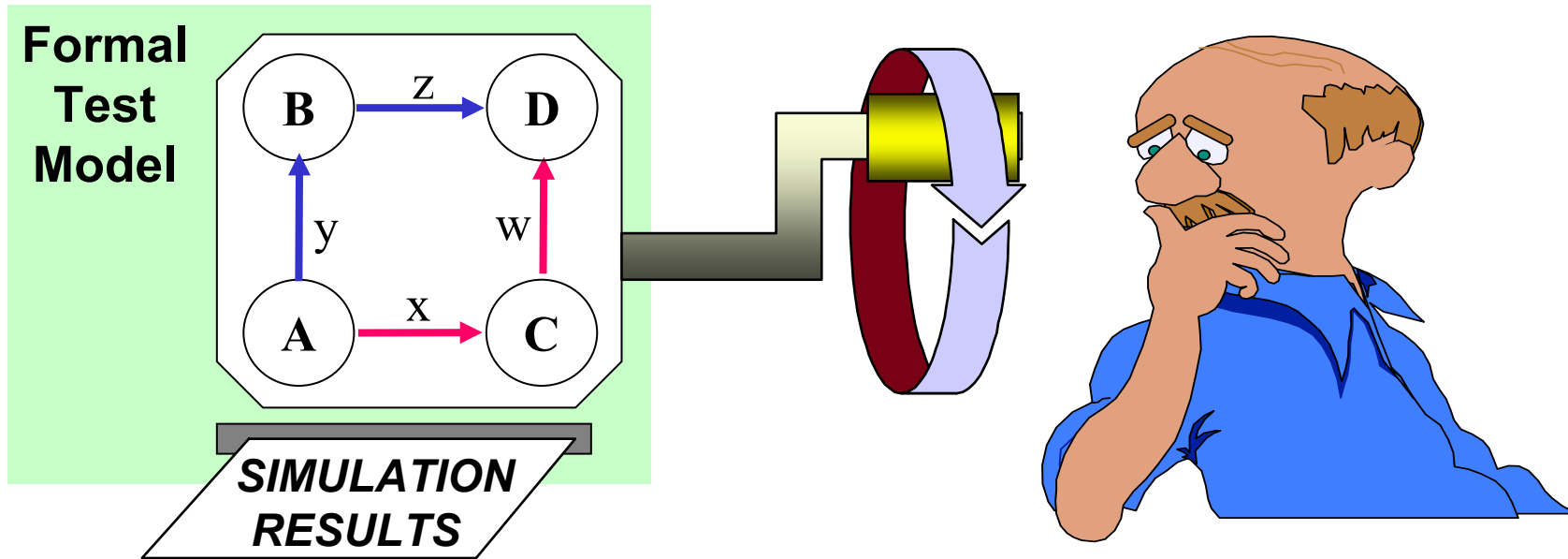
- Test scripts manually prepared
- Time consuming
- Incomplete
- Informal
- Opportunity to introduce defects



Auto Test Generation Process



Formal Test Models - Benefits



- Verification and validation – simulation
- Higher level of abstraction
- Test model maintained only
- Model may form part or all of the requirements

Benefits of Auto Test Generation

Reduced Cost of Producing Test Suites

- Hand-written tests normally lead to 'representative' tests.
- Greater *get off the ground* cost to set up test model
- Reported reduction factors of up to 3x



Greater Test Coverage

- Exhaustive nature of auto-testing tools

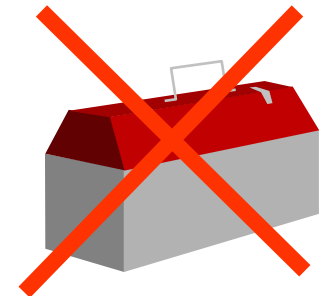
Improved Test Quality

- All possible scenarios are explored thereby generating unexpected or obscure tests



Easier Test Suite Maintenance

- Only the test model needs to be maintained, rather than the test suite

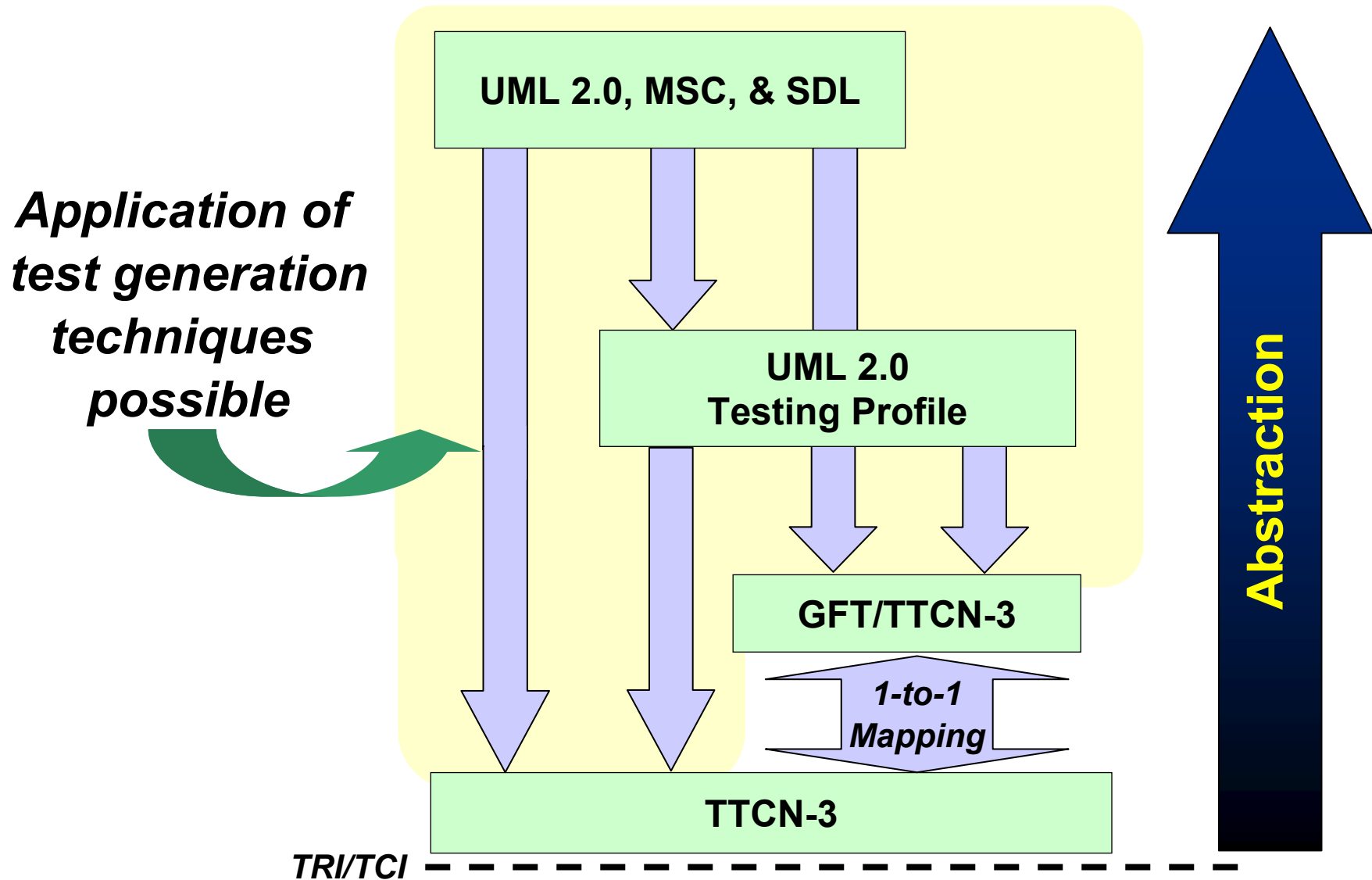


Some Test Generation Considerations

- **Abstraction**
- **Complexity**
 - Constraint mechanisms
 - Selection strategies
- **Test strategies/coverage**
- **Data**
 - Static and dynamic
 - Data Pools
- **Model/test correctness**
 - Valid tests
- **Test configuration**
- **Traceability**



TTCN-3 Landscape



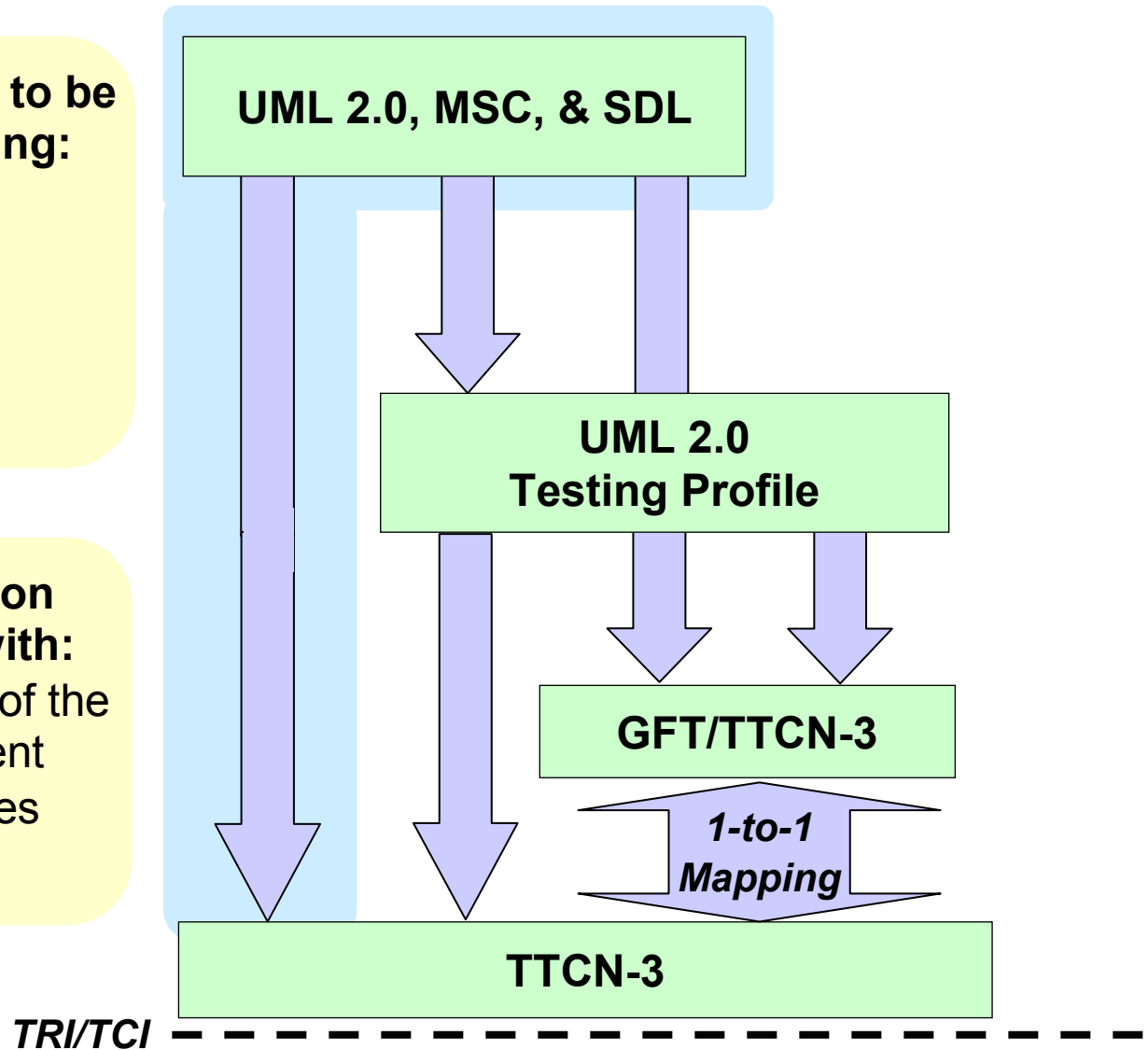
Abstraction (1)

The user does not need to be concerned with specifying:

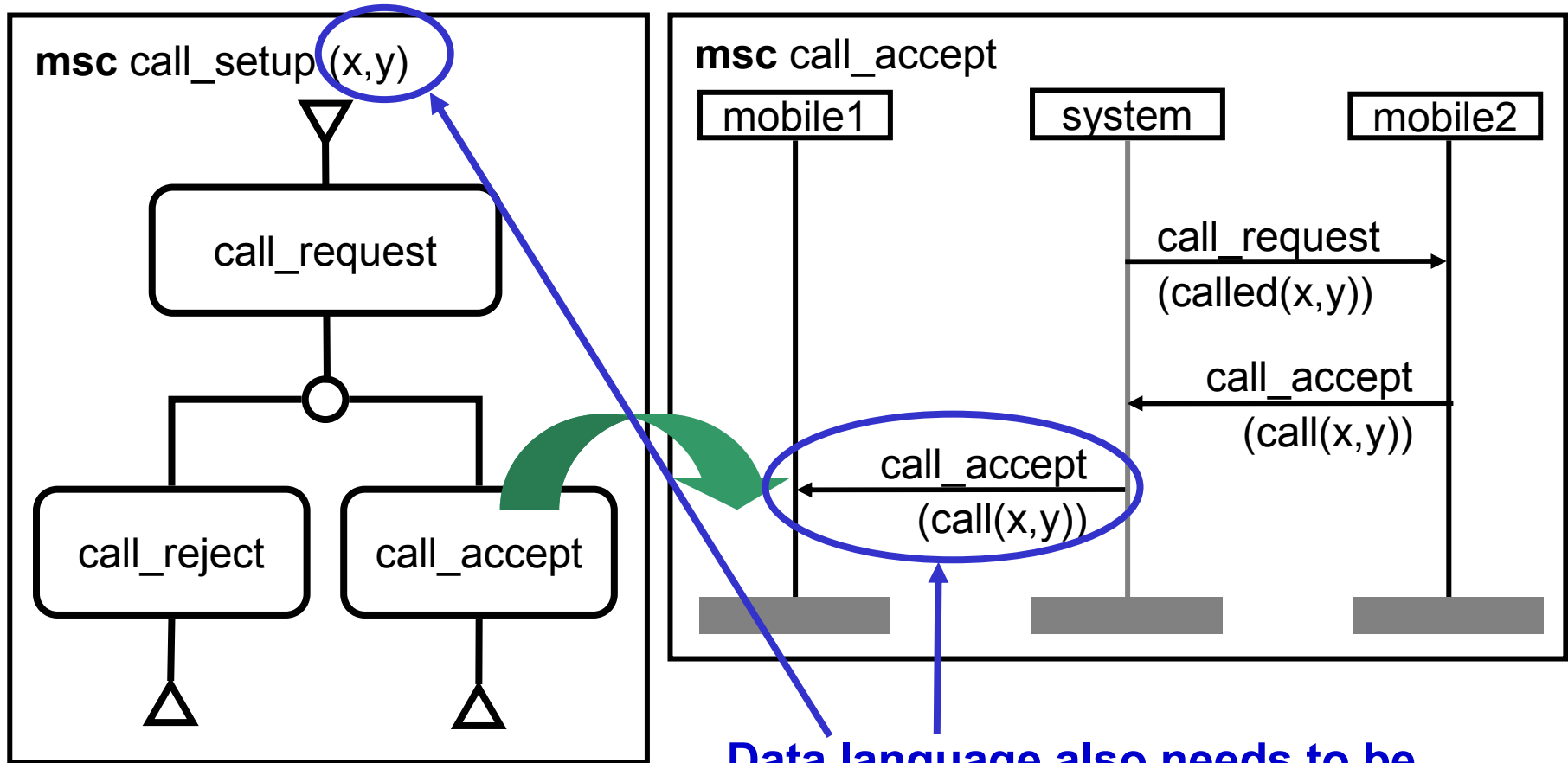
- test cases
- test configuration
- verdict handling
- exception handling
- test control

At this level of abstraction the user is concerned with:

- the behaviour definition of the system with its environment
- test coverage & strategies
- model correctness

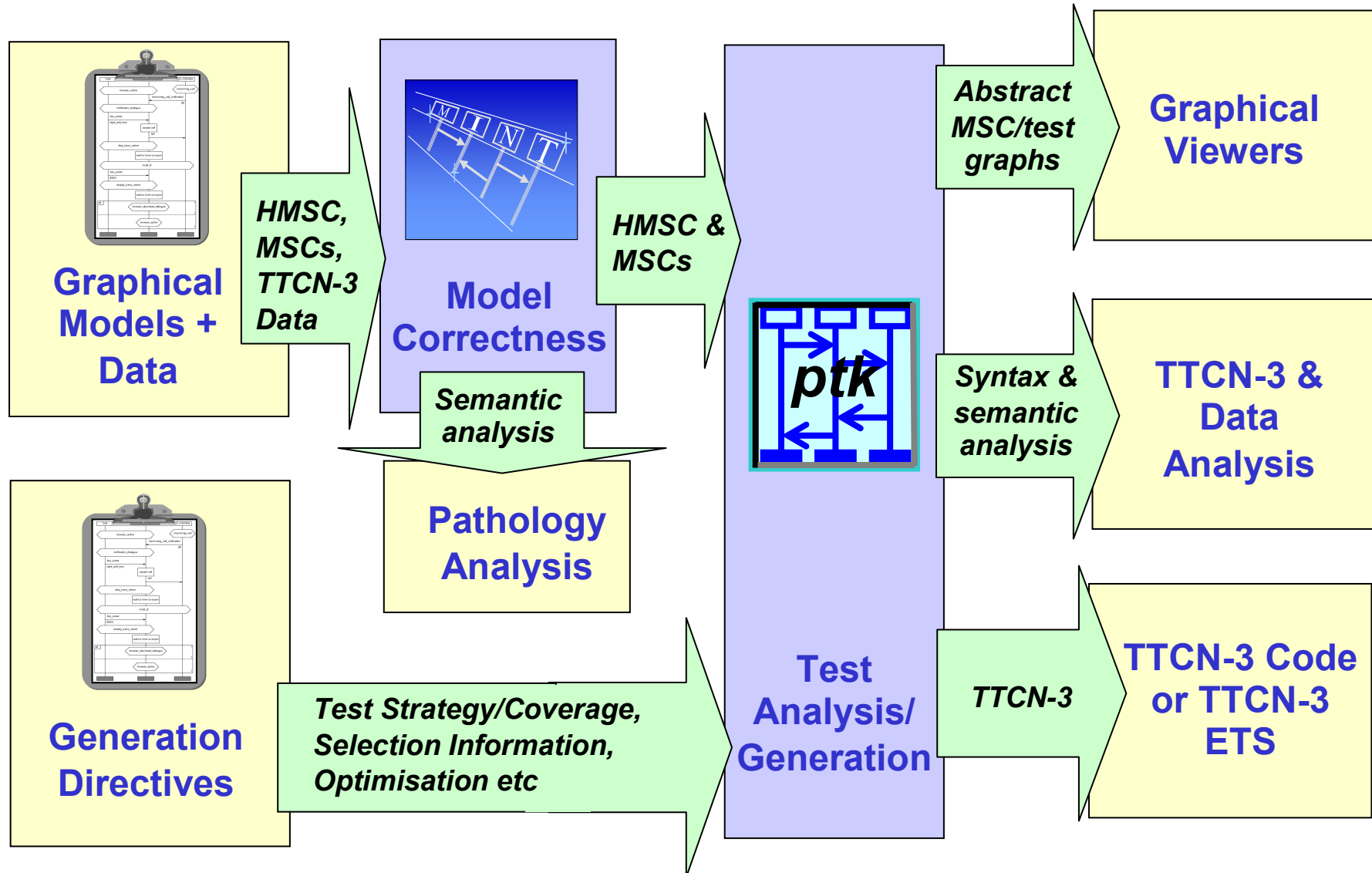


Simple TTCN-3 Example



Data language also needs to be considered. In this case MSC is parameterised with TTCN-3.

Example: Test Generation Process



Example: Model Correctness/Test Analysis

- **Model semantics are represented in terms of traces**
 - Semantic analysis = 26 traces
 - Test analysis = 5 test traces
- **Model correctness identifies potential errors in the specification that can lead to invalid tests.**

The image shows two overlapping windows. The left window is a command prompt titled "C:\WINDOWS\system32\cmd.exe" running the ptk 3.7 tool. The right window is "daVinci Presenter Modeler 3.0.5 - TTCN_example_ta" displaying a state transition graph.

```
ptk 3.7
Copyright (c) Motorola 2003
Built on Tue Dec 9 14:16:58 GMTST 2003

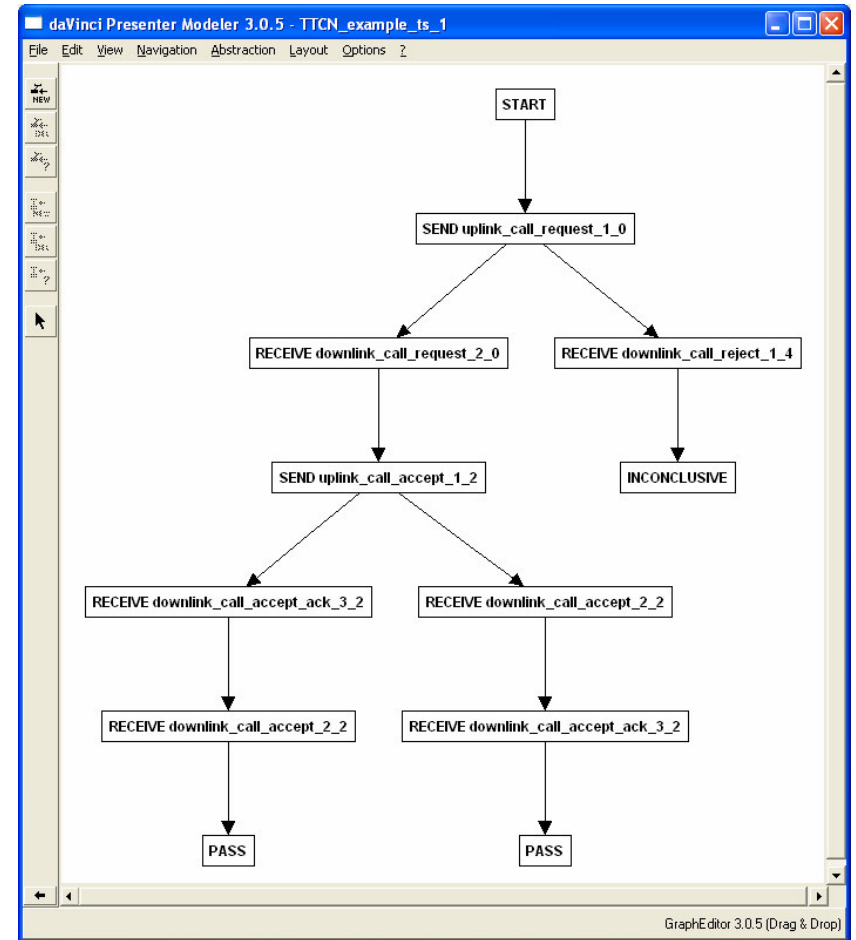
Parsing "C:\motorola\ptk\demos\ttcn3\TTCN_example.mpr"
Parsing "C:\motorola\ptk\demos\ttcn3\call_request.mpr"
Parsing "C:\motorola\ptk\demos\ttcn3\call_accept.mpr"
Parsing "C:\motorola\ptk\demos\ttcn3\call_reject.mpr"
ERROR: C:\motorola\ptk\demos\ttcn3\call_accept.mpr:6.13-
in non-local choice with SEND downlink_call_reject_1_4.
Warning: C:\motorola\ptk\demos\ttcn3\call_reject.mpr:10.
ject_ack" is a blocked message which is RESOLVABLE, it i
\ptk\demos\ttcn3\call_request.mpr:9.12-33: "downlink_cal

MSC pathology summary (reporting checked pathologies only)
resolvable blocked      : 1
irresolvable blocked    : 0
non-local choice        : 1
non-local ordering      : 0
false underspecification : 0
C:\motorola\ptk\demos\ttcn3>
```

The state transition graph in the right window shows a hierarchical structure of states. It starts with a root state, which branches into two states. The left branch further divides into two states, each of which then branches into two more states. The right branch from the root also divides into two states, each of which then branches into two more states. The graph ends with several terminal states represented by small squares.

Example: TTCN-3 Test Generation

- Depending upon test generation directives a complete TTCN-3 suite can be generated.
- In this case we:
 - Trade off path coverage vs no of test cases => 2 test cases
 - Only one test case may need to be executed => test control strategy
 - Useful tests produced
- Generate TTCN-3 code
 - Conformance tests
 - Concurrent test scripts
 - Load tests

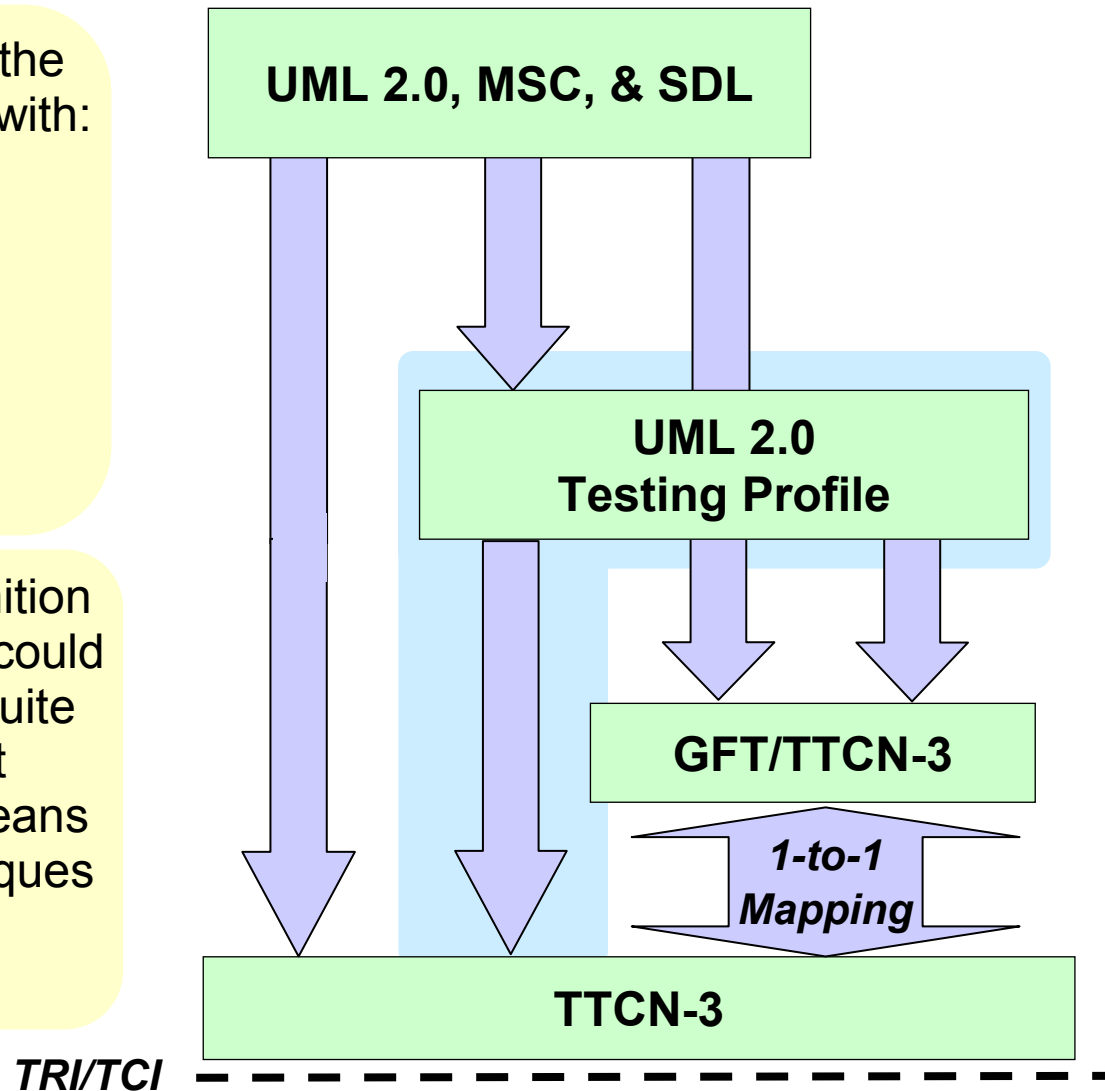


Abstraction (2)

At this level of abstraction the user is usually concerned with:

- test cases
- test configuration
- verdict handling
- exception handling
- test control
- specification correctness

However, the level of definition can vary. For example, it could be that only a partial test suite is defined containing a test configuration only. This means that test generation techniques can be applied for missing aspects.



Thank you

Almost lunch time...

– Questions?