Model-based testing tools

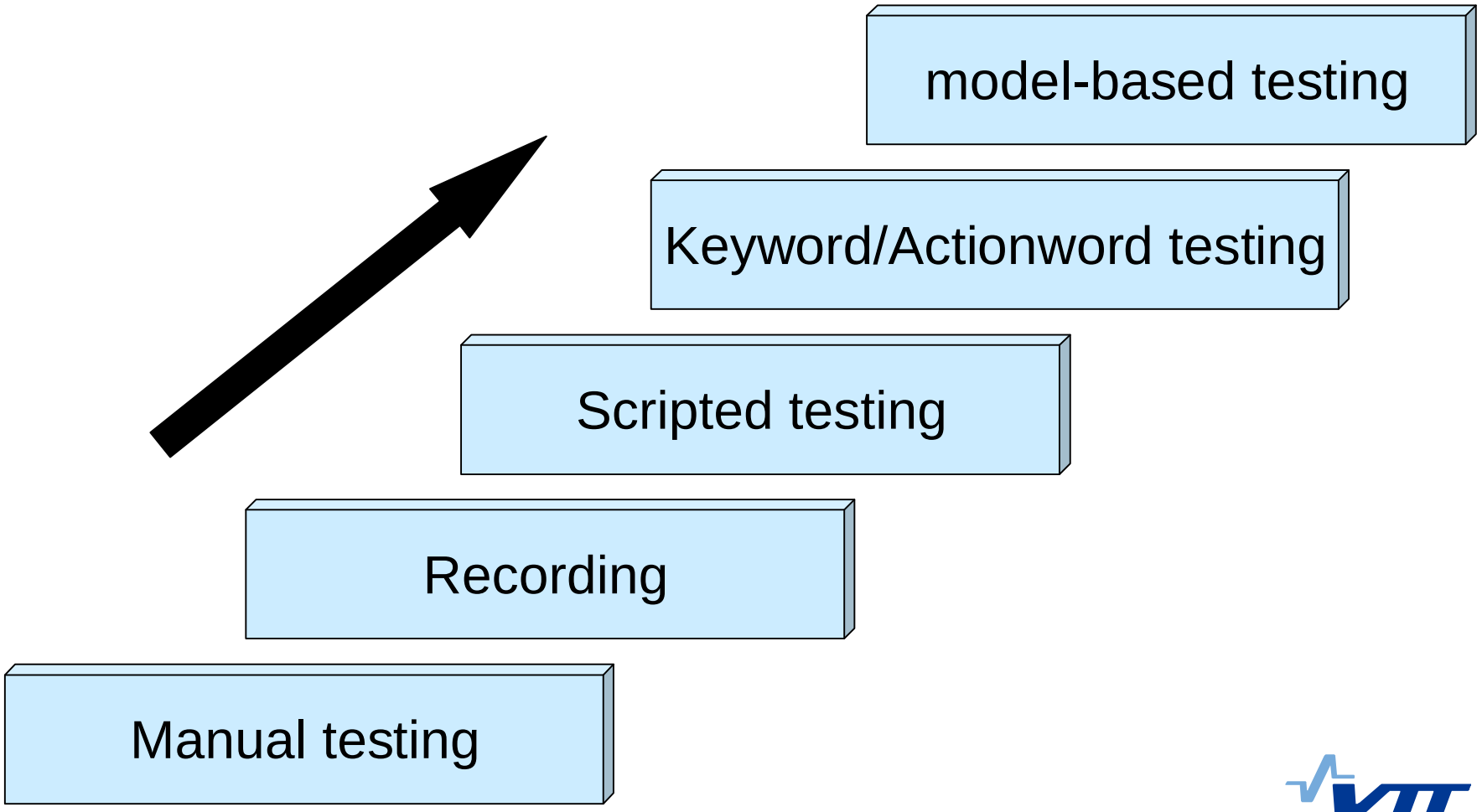
Olli-Pekka Puolitaival

Business from technology

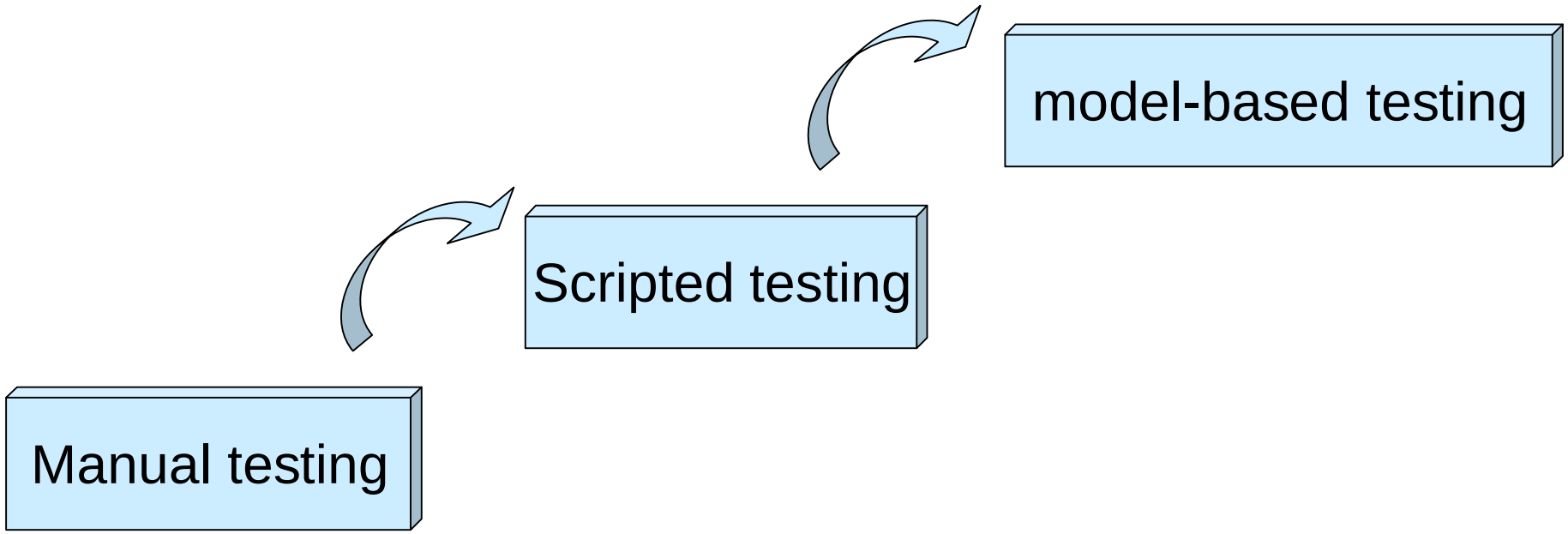
Index

1. Software testing evolution
2. model-based testing (MBT): main idea
3. MBT: step by step
4. MBT: tools
5. Questions

Software testing evolution

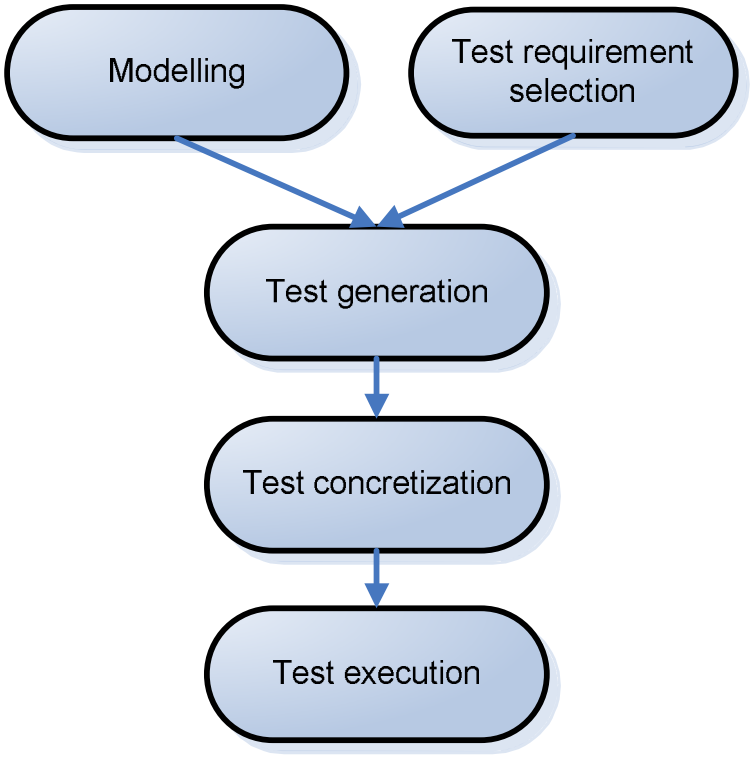


Software testing evolution



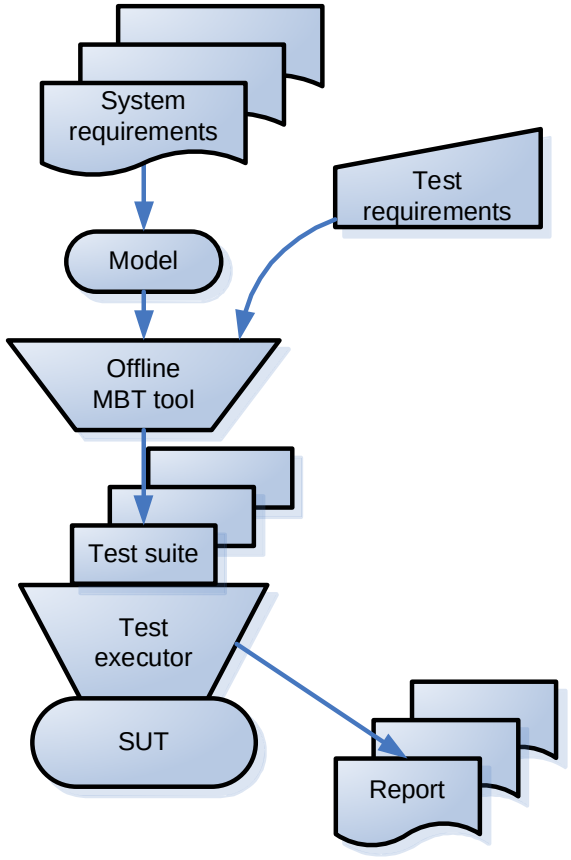
Picture adopted from: http://www.tol.oulu.fi/projects/wg4/minutes/WG4_0507_TTY.pdf

What is model-based testing?



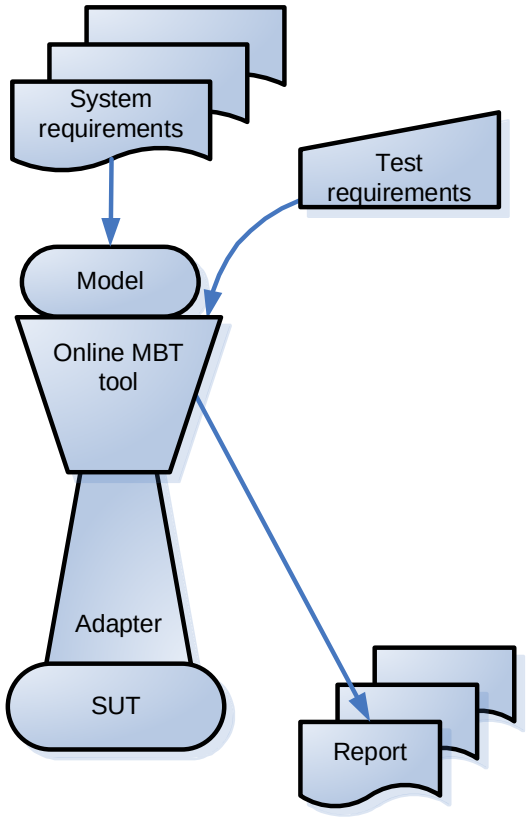
- **Model-based testing** is software testing in which test cases are generated in whole or in part from a model that describes some (usually functional) aspects of the system under test (SUT)
- **Almost synonyms**
  - Model-driven testing
  - Test generation
  - Hardware in the loop
- **Two main approach**
  - Online MBT
  - Offline MBT

Offline MBT

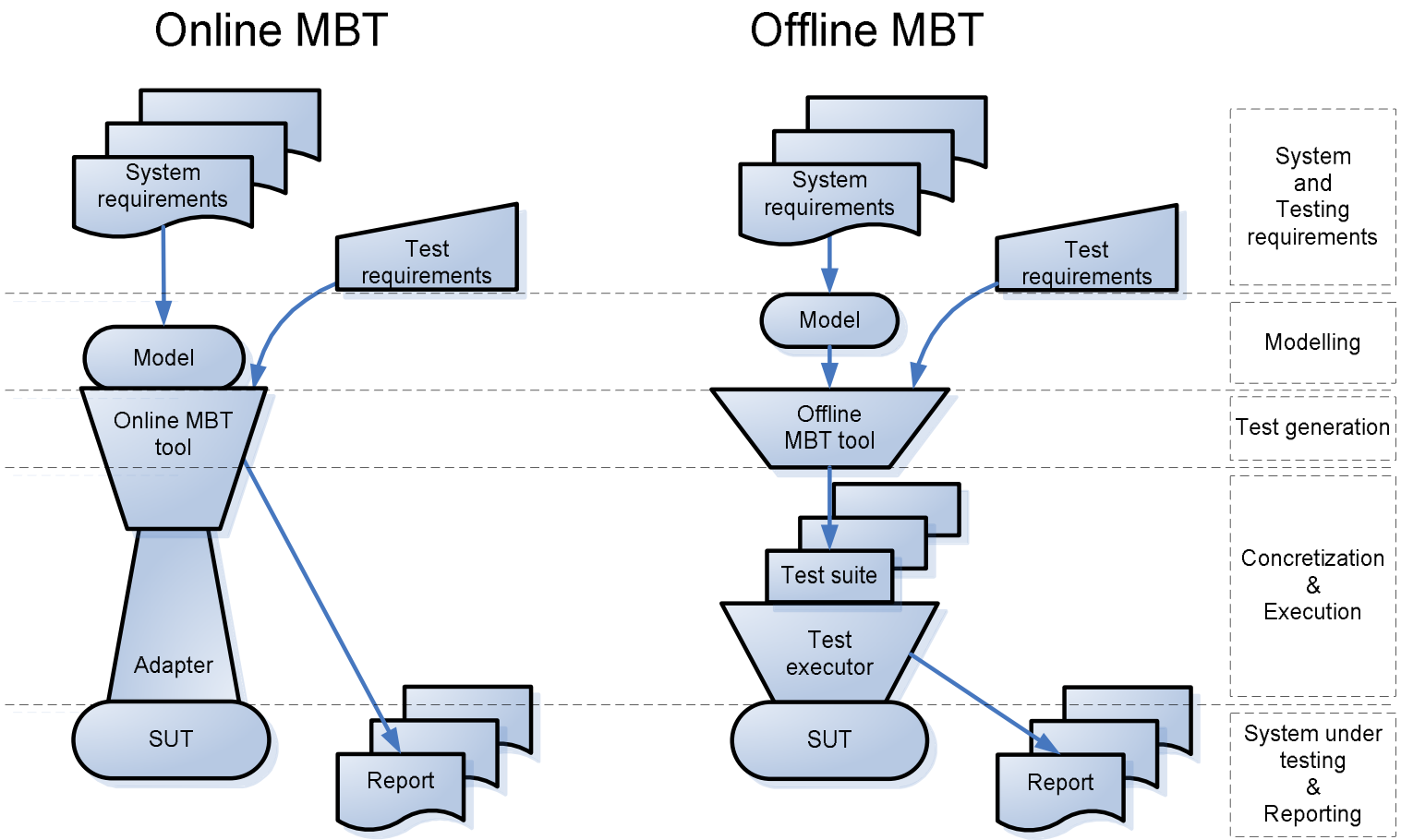


- Automate test case generation
- Offline MBT means generating a finite set of tests and execute those later
- This allows automatic test execution in third party test execution platform
- Makes possible to create a tool chain:

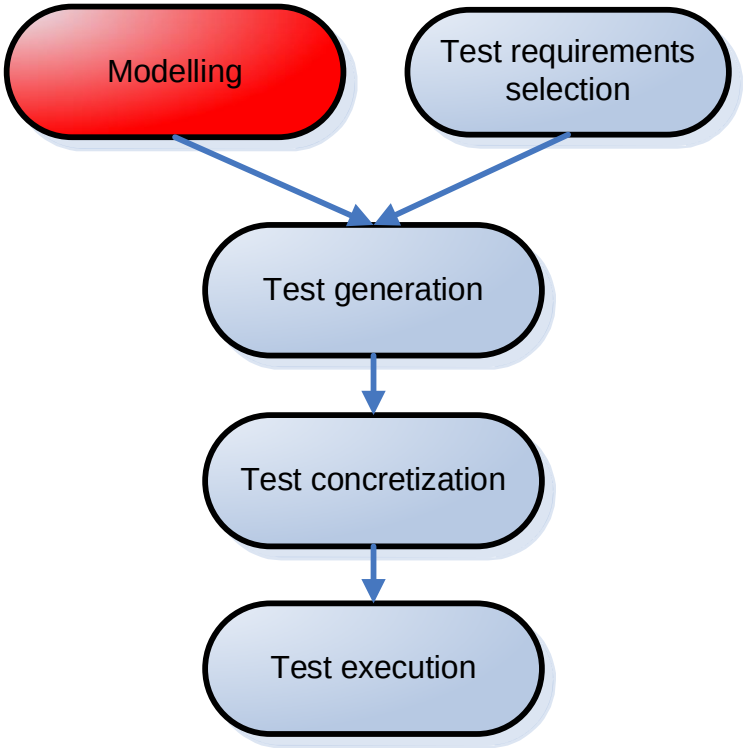
Online MBT



- Test case generation and execution in motion
  - Next step is design after the output receiving
- Testing nondeterministic systems
- Infinite test suite running

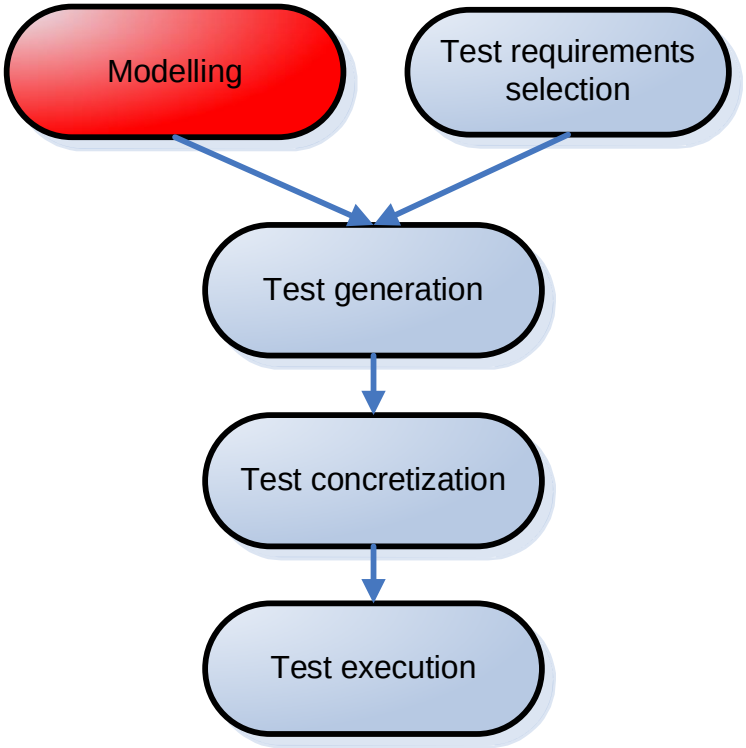

Online MBT
System requirements
Test requirements
Model
Online MBT tool
Adapter
SUT
Report
Offline MBT
System requirements
Test requirements
Model
Offline MBT tool
Test suite
Test executor
SUT
Report
System and Testing requirements
Modelling
Test generation
Concretization & Execution
System under testing & Reporting

Modelling (1)

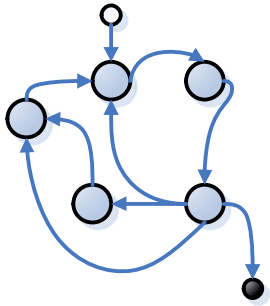


- **Purpose:** To describe the system requirements for test generator
- Many faults can be found already in this phase
- **Important**
  - High abstraction level
  - The model includes also expected output of the SUT
- **Two main aspects**
  - Design model
  - Test model

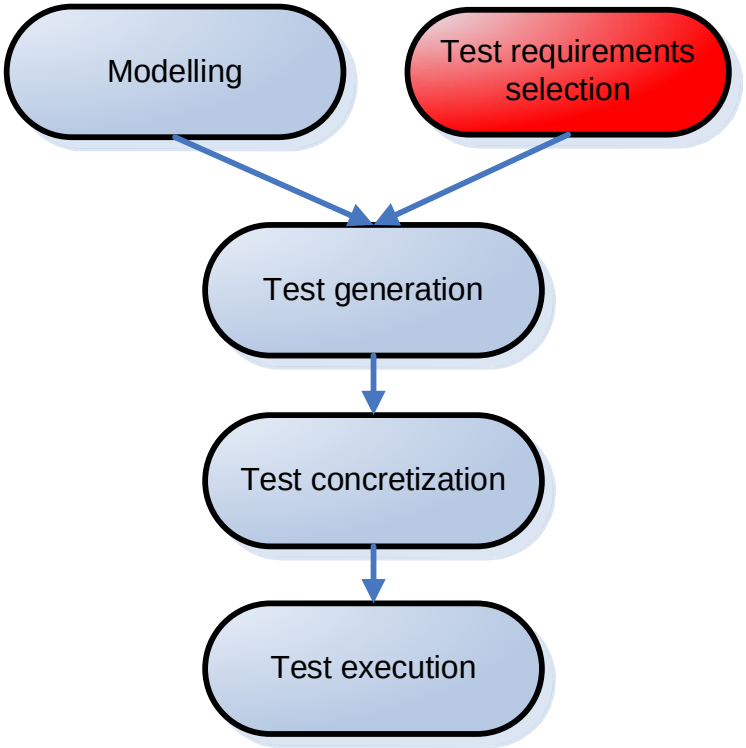
Modelling (2)



- **Choosing criteria:**
  - **Modelling notations**
    - General vs. domain-specific
    - Control oriented vs. data flow oriented
  - **Model validation**
    - Syntax
    - Behaviour
- **Example:**

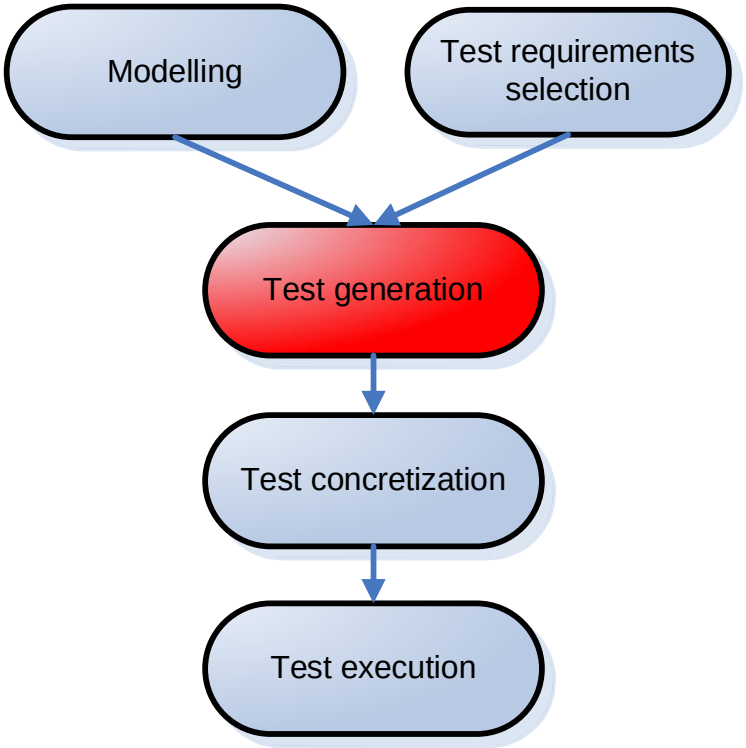


Test requirements selection



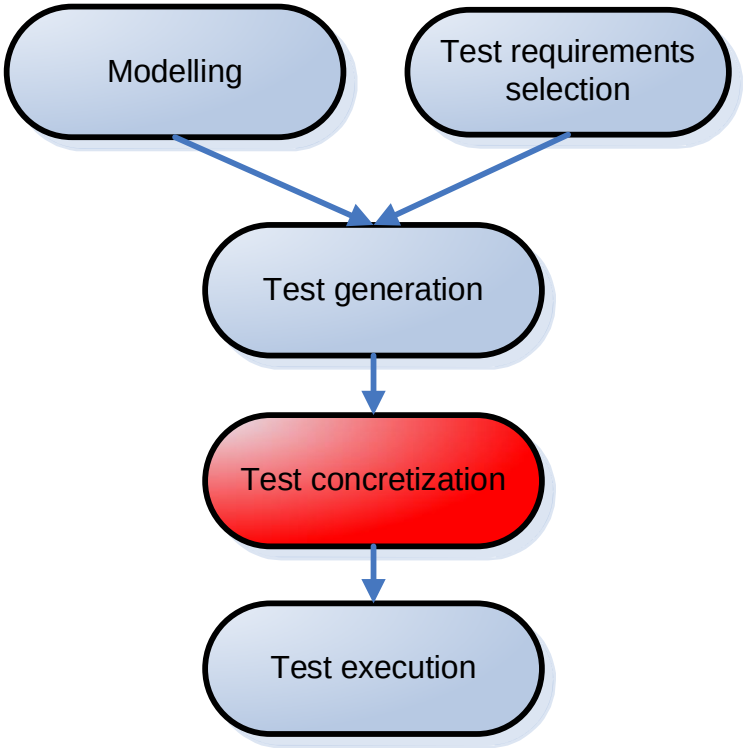
- **Purpose:** guiding test generation
- **Three main categories:**
  1. Targets in the model
  2. Coverage criteria
     - State coverage
     - Transition coverage
     - Not means code coverage
  3. Walking algorithms
     - Random walking
     - Coverage guided

Test generation

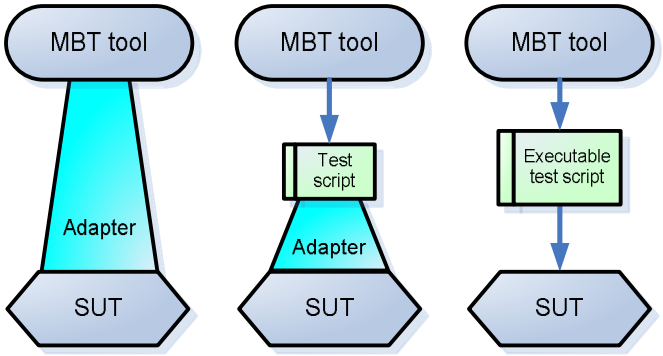


- **Purpose:** Design as well tests as possible
- **Offline**
  - Searching algorithms
  - Test are written in determined format
- **Online**
  - Walking algorithms or light searching algorithms
  - The next test step is decided after the previous one execution and output value receiving
  - Algorithms have to be fast

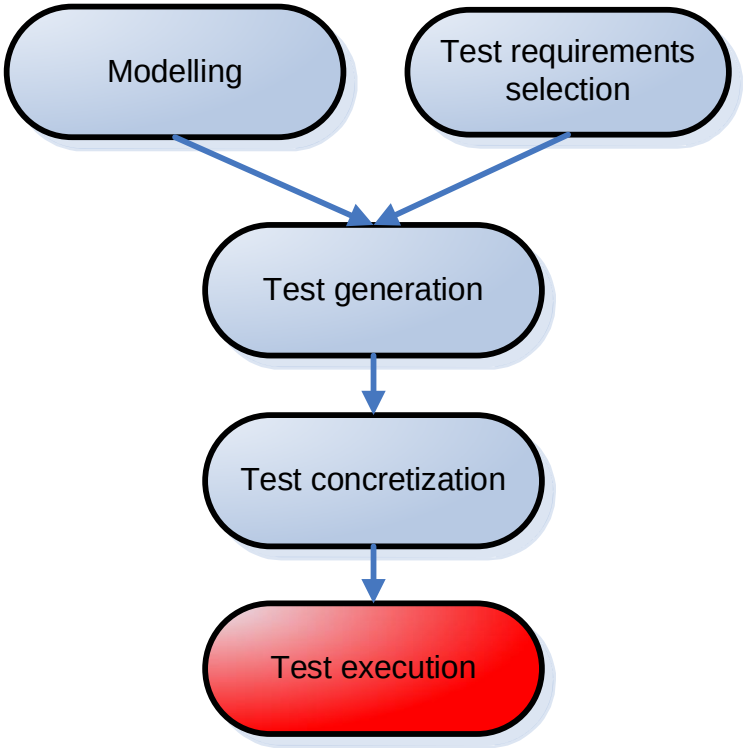
Test concretization



- **Purpose:** Concretize abstract test suite to executable level
- **Tools are providing**
  - Various test exporting formats
  - Do-it-yourself plug-ins
- **Three concretization example:**

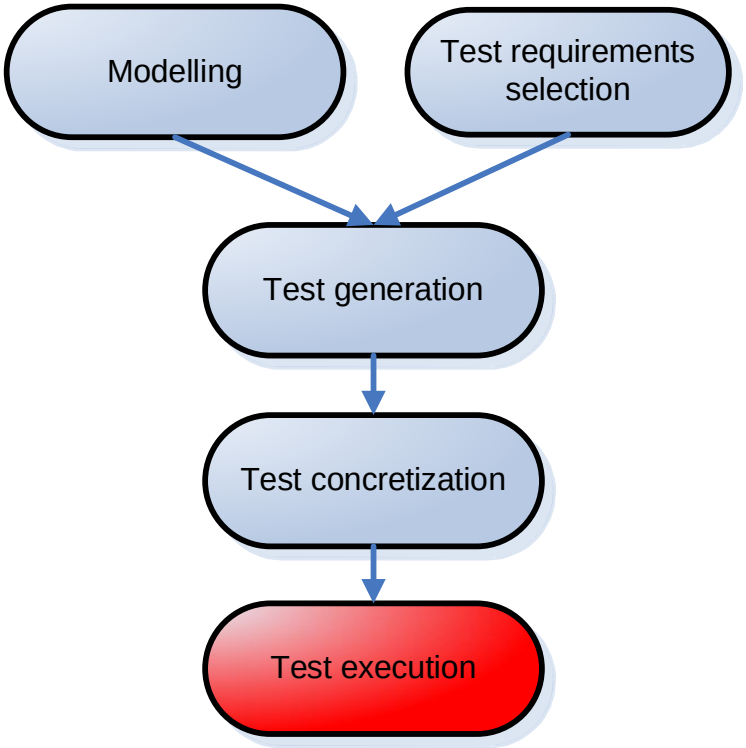


Test execution (1)



- **Purpose:** Test executing and output comparing to the expected output
- **Offline**
  - Test are run in external test execution platform
  - The platform is writing the document and analysing the results

Test execution (2)



- **Online**
  - Test are executed during the test designing
  - MBT tool write a report and analyse the results
  - Makes possible to handle SUT nondeterministic
- **Analysing results**
  - Traceability
  - Reporting

Main benefits of MBT

- **Easier test suite maintenance**
- **Automated test design**
  - Save effort
- **Better test quality**
  - no human faults
  - Computer can find more combinations for complex systems than human brain
- **Online MBT provides also**
  - Testing nondeterministic systems
  - Infinite test suite

MBT tools analyse

- **Purpose:** To describe most relevant available MBT tools as example
- **Tools:**
  - Leirios test generator
  - MaTeLo
  - Qtronic
  - Reactis
  - Spec Explorer

Leirios Test Designer (1)

- **Operating system**: Windows, Linux (Ubuntu)
- **Offline/Online**: Offline
- **Modelling**:
  - Languages: UML+OCL
  - No internal modeller
  - Supporting third party modelling tools:
    - Rational Software Modeler
    - Borland Together
- **Model validation**
  - Checking generated test cases

Leirios Test Designer (2)

- **Test generation guiding**
  - Targets: test are consisting three parts:
    - Preamble
    - Body (=target)
    - Postamble
- **Test writing**:
  - Exporting test suite
  - Large variety of exporting formats
  - Do-it-yourself adapter is provided
- **Report & traceability**
  - Traceability matrix

MaTeLo (1)

- **Operating system**: Windows
- **Offline/Online**: Offline
- **Modelling:**
  - Language: Markov Chain model (FSM + transition probabilities, extended with variables, Scilap/Scicos and Matlab/Simulink functions
  - Include a modelling tool
  - Supporting third party modelling tools:
    - UML in XMI format
    - MSC-PR from SDL tool
- **Model validation**
  - Manually checking the generated HTML-file

MaTeLo (2)

- **Test generation guiding**
  - Coverage criteria
    - Boundary value
    - State and transition coverage
  - Others
    - Most probable route
    - Random
- **Test writing**:
  - Exporting formats:
    - TTCN-3
    - HTML
    - TestStand
- **Extra:**
  - Report management
  - Asynchronous inputs

Conformiq Qtronic (1)

- **Operating system**: Windows or Linux
- **Offline/Online**: Both
- **Modelling**
  - Languages:
    - CQλ (variant of LISP)
    - QML (UML Stateflow + variant of java)
  - Include a modelling tool
  - Input supporting formats:
    - UML2.0/2.1 models in XMI2.0/2.1 format
- **Model validation**
  - Test suite exporting as HTML sequence diagram

Conformiq Qtronic (2)

- **Test generation guiding**
  - 9 coverage criteria
  - Targets (called as requirements)
  - 3 walking algorithms for online MBT
- **Test writing (offline)**:
  - Export test suite with plug-ins (HTML, TTCN-3)
  - Do-it-yourself plug-ins are provided
- **Test execution (online)**
  - Adapter plug-in for test executing
  - Logger plug-in for logging
- **Extra:**
  - Provide nondeterministic systems testing
  - Search depth control, various test stopping criteria

Reactis (1)

- **Operating environment**: Windows with Matlab+ Simulink
- **Offline/Online**: Offline
- **Modelling**:
  - No internal modeller
  - Support model created by Matlab/Simulink/Stateflow
- **Model validation:** Simulating

Reactis (2)

- **Test generation guiding**
  - Random
  - 10 coverage criteria
- **Test execution**:
  - Exporting formats
    - Matlab formats
    - Plain ASCII
    - Comma separate value
- **Report & Traceability**
  - The test executor is reporting

Spec Explorer (1)

- **Operating system**: Windows
- **Strongly bounded with:** Visual Studio
- **Offline/Online**: Both
- **Modelling language**:
  - Spec# (extended C#)
  - AsmL (Abstract State Machine Language)
  - FSM for visualization
- **Modeller**: MS Word or some other text editor

License only for research

Spec Explorer (2)

- **Test generation guiding:**
  - Random walking
  - Shortest path
  - Transition coverage
- **Test writing (offline)**:
  - Xml

Further reading

- Practical Model-Based Testing a tool approach, Mark Utting and Bruno Legeard

Thank you

- Olli-Pekka Puolitaival
- Research trainee
- Automated Testing Platforms team, VTT
- olli-pekka.puolitaival@vtt.fi
- +358 40 060 6293