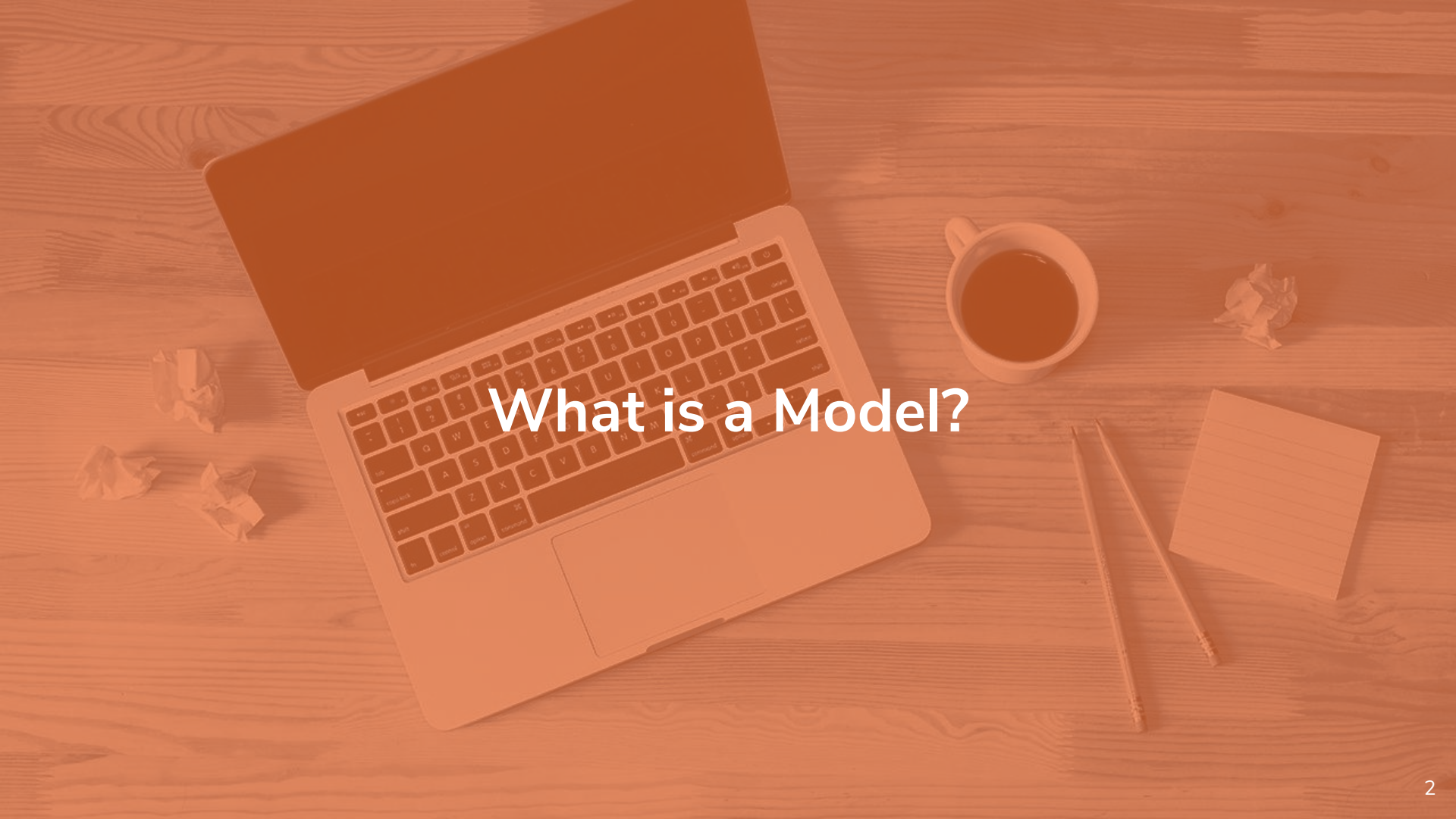
Universidade do Porto
Faculdade de Engenharia
FEUP

Model Based Testing

Alexandre Moreira
Pedro Carvalho



What is a Model?

Model

Model: description of a system's behavior

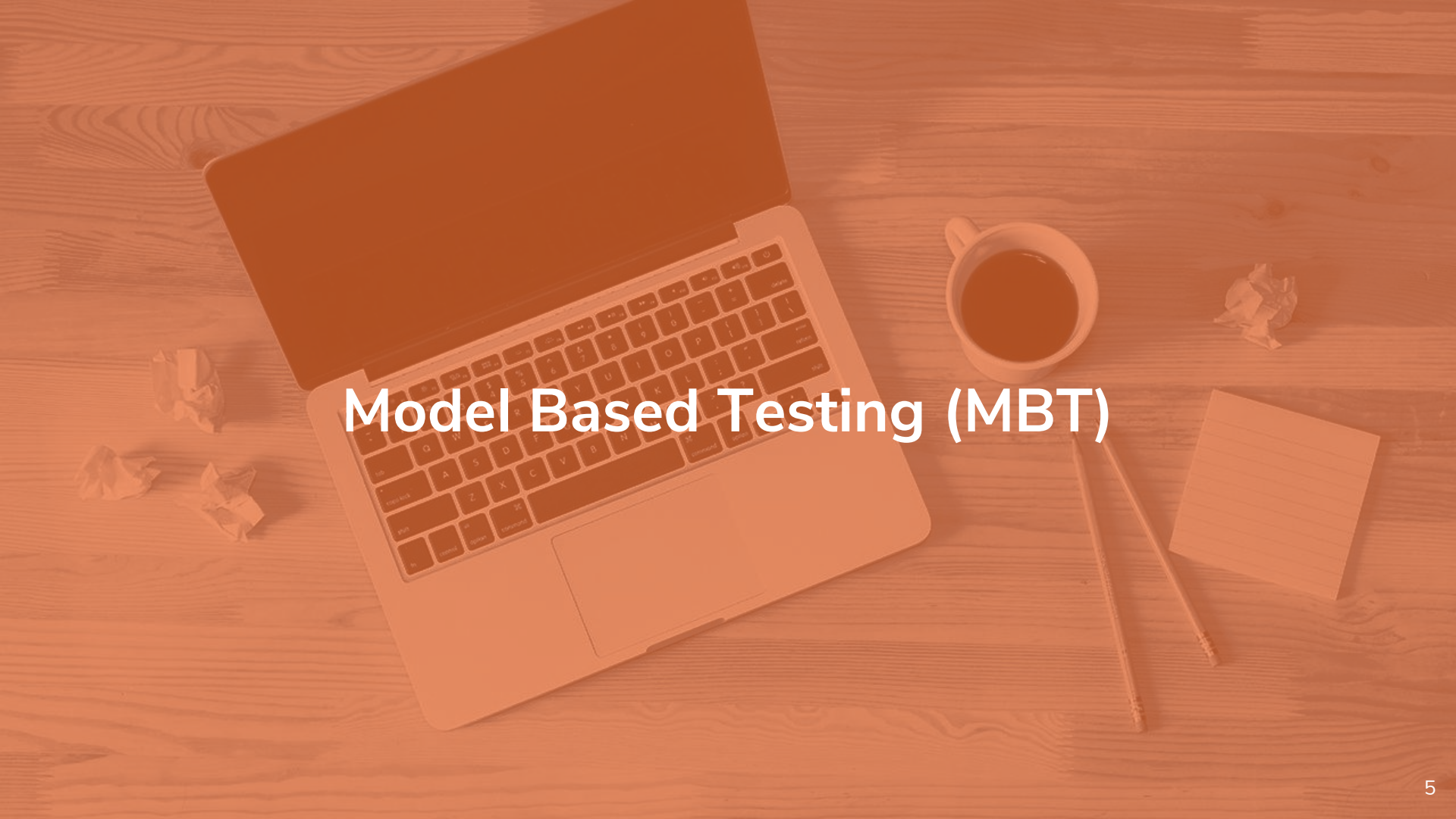
Behavior: can be described in terms of input sequences, actions, conditions, output and flow of data from input to output. It should be practically understandable and can be reusable

Model

There are many different types of models available and they all describe different aspects of the system behavior

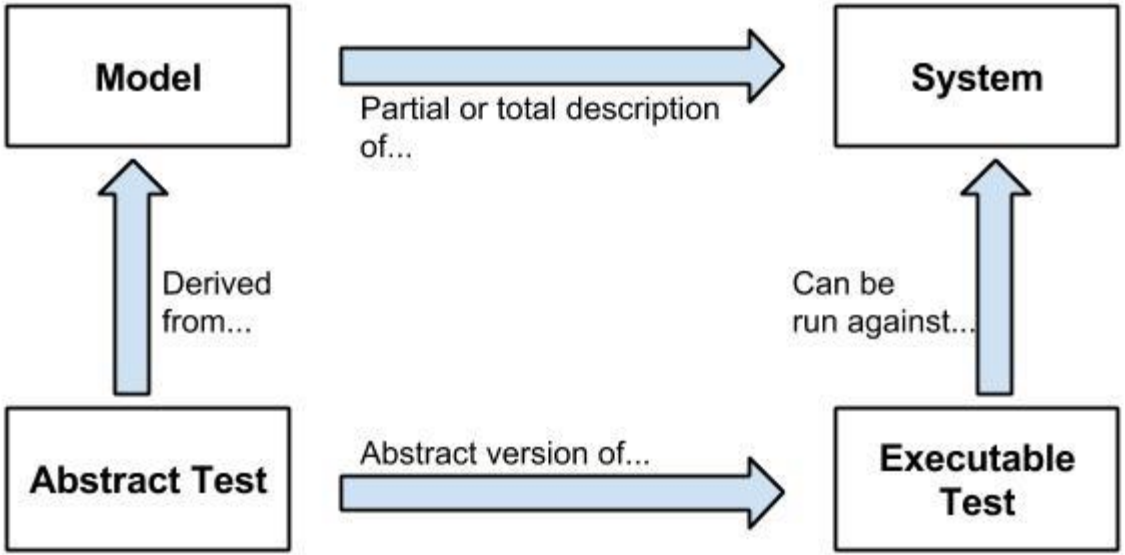
Examples:

- Data Flow
- Control Flow
- Dependency Graphs
- Decision Tables
- State transition machines



Model Based Testing (MBT)

Model Based Testing



Model Based Testing

It is a testing technique in which test cases are derived from a model that describes the functional aspects (behaviours) of the system under test (SUT)

Basically, the idea is that from a model or semi-formal model, complete test cases can be generated

Model Based Testing

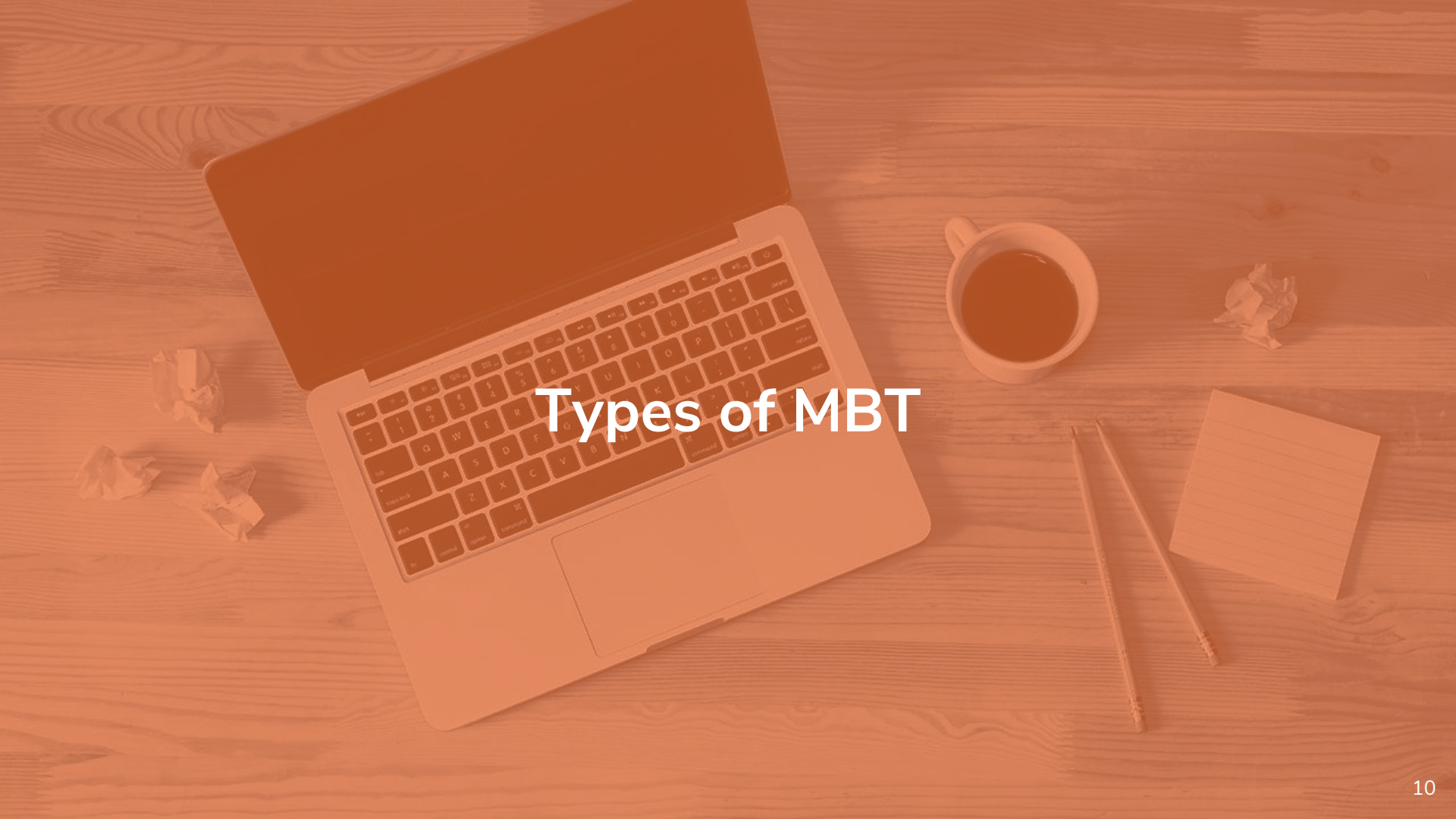
Not only:

- does it allow to evaluate requirements independently of algorithm design and development
- but it also helps automate other verification tasks and streamlines the review process by linking test cases and verification objectives to high-level test requirements

Model Based Testing

Two very important aspects:

- can be applied to both hardware and software testing
- includes both offline and online testing



Types of MBT

Types of MBT

Offline (a priori): generation of Test Suites before test execution

Online (on-the-fly): generation of Test Suites during test execution

Test suite: collection of test cases

Types of MBT

Offline:

- automates test case generation
- it means generating a finite set of tests and execute those later
- it automatic test execution in third party test execution platform

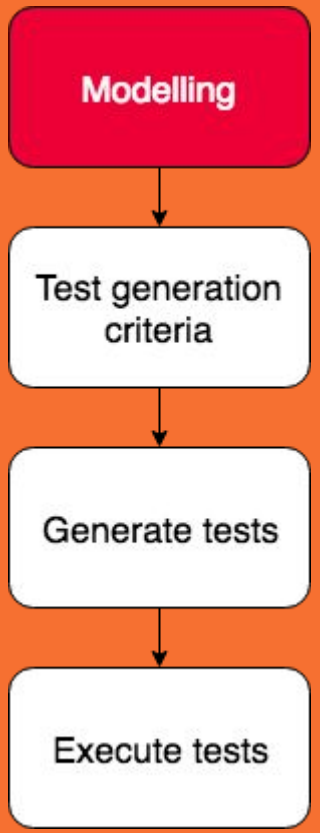
Types of MBT

Online:

- test case generation and execution in motion
- next step is design after the output receiving
- testing nondeterministic systems
- infinite test suite running

Steps of MBT

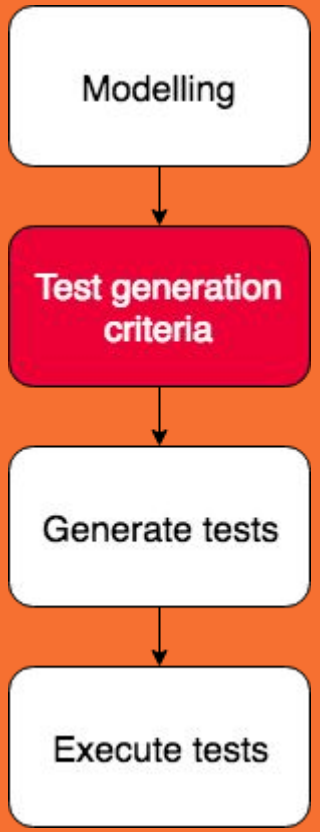
Steps of MBT



Modelling:

- formalize the control points and observation points of the system, its expected dynamic behaviour, the business entities associated with the test, and some data for the initial test configuration
- elements such as transitions or decisions are linked to the requirements, in order to ensure bi-directional traceability between the requirements and the model, and later to the generated test cases

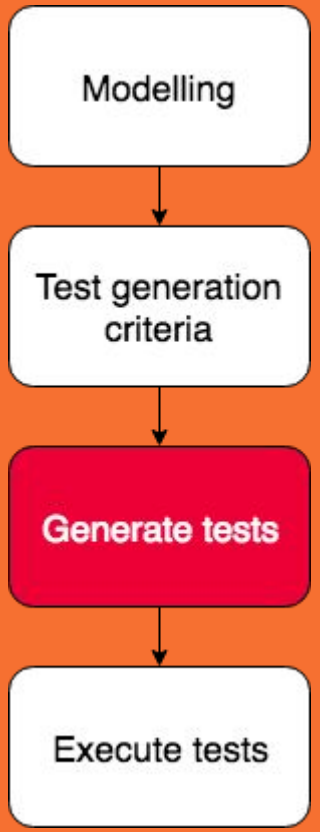
Steps of MBT



Test generation criteria:

- select the highest priority tests, or to ensure good coverage of the system behaviour
- based on structural model coverage, using well known test design strategies such as equivalence partitioning, cause-effect testing, pair-wise testing, process cycle coverage, or boundary value analysis
- ensures that the generated test cases cover all the requirements, perhaps with more tests generated for requirements that have a higher level of risk

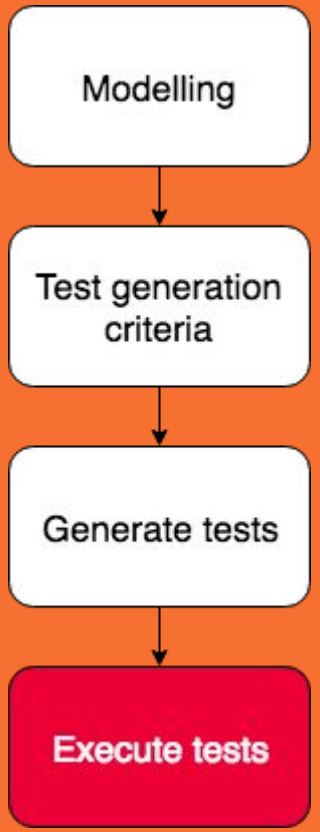
Steps of MBT



Generate tests:

- fully automated process
- typically a sequence of high-level SUT actions
- easily understandable by humans and are complete enough to be directly executed on the SUT by a manual tester
- tests generated from the test model may be structured into multiple test suites

Steps of MBT

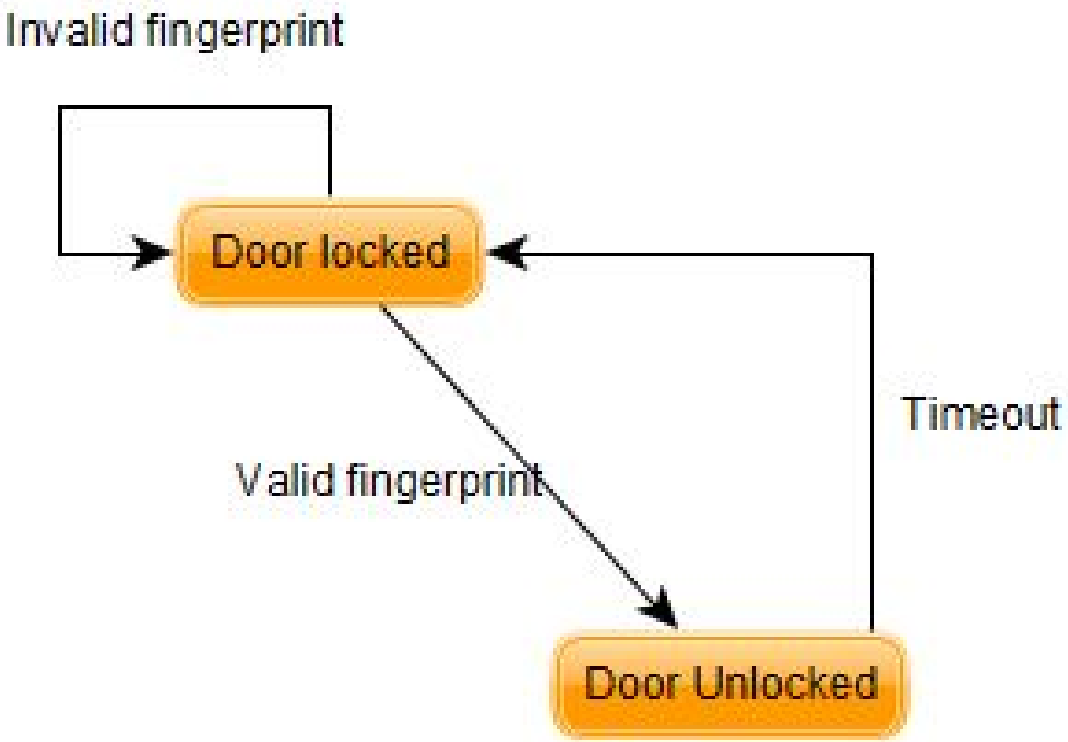


Execute tests:

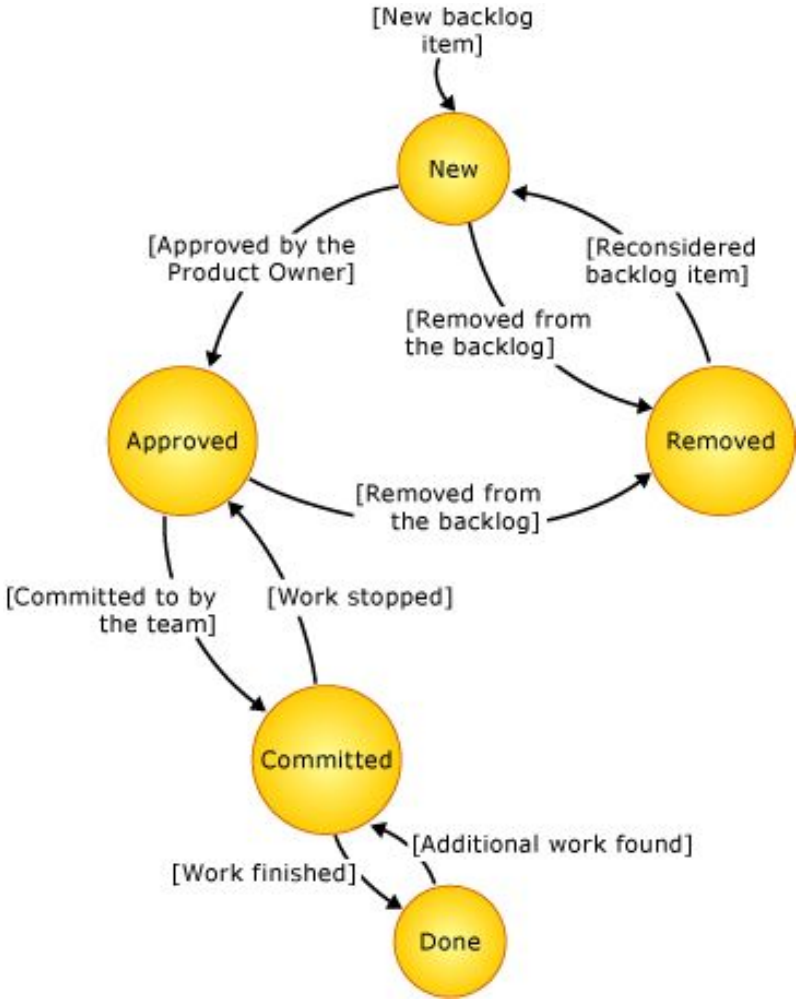
- executed either manually or within a standard automated test execution environment
- we find that some tests pass and some tests fail
- failing tests indicate a discrepancy between the SUT and the expected results designed in the test model
- which then needs to be investigated
- good at finding SUT errors
- highly effective at exposing requirements errors

Examples

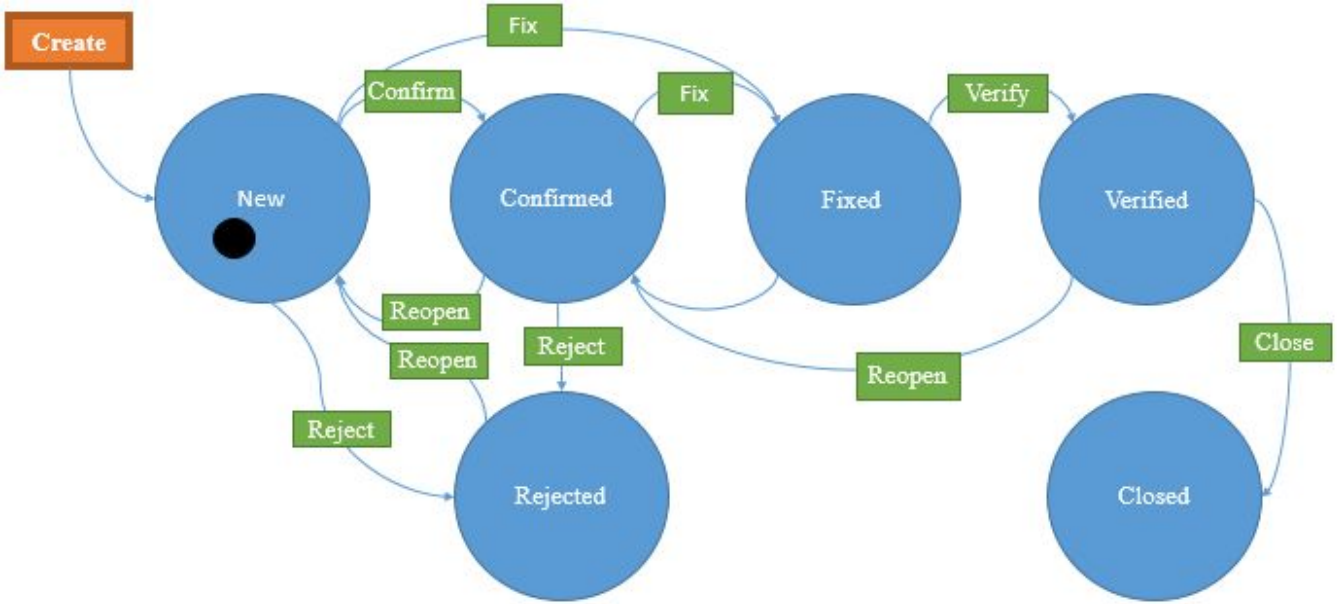
Examples


Invalid fingerprint
Door locked
Timeout
Valid fingerprint
Door Unlocked

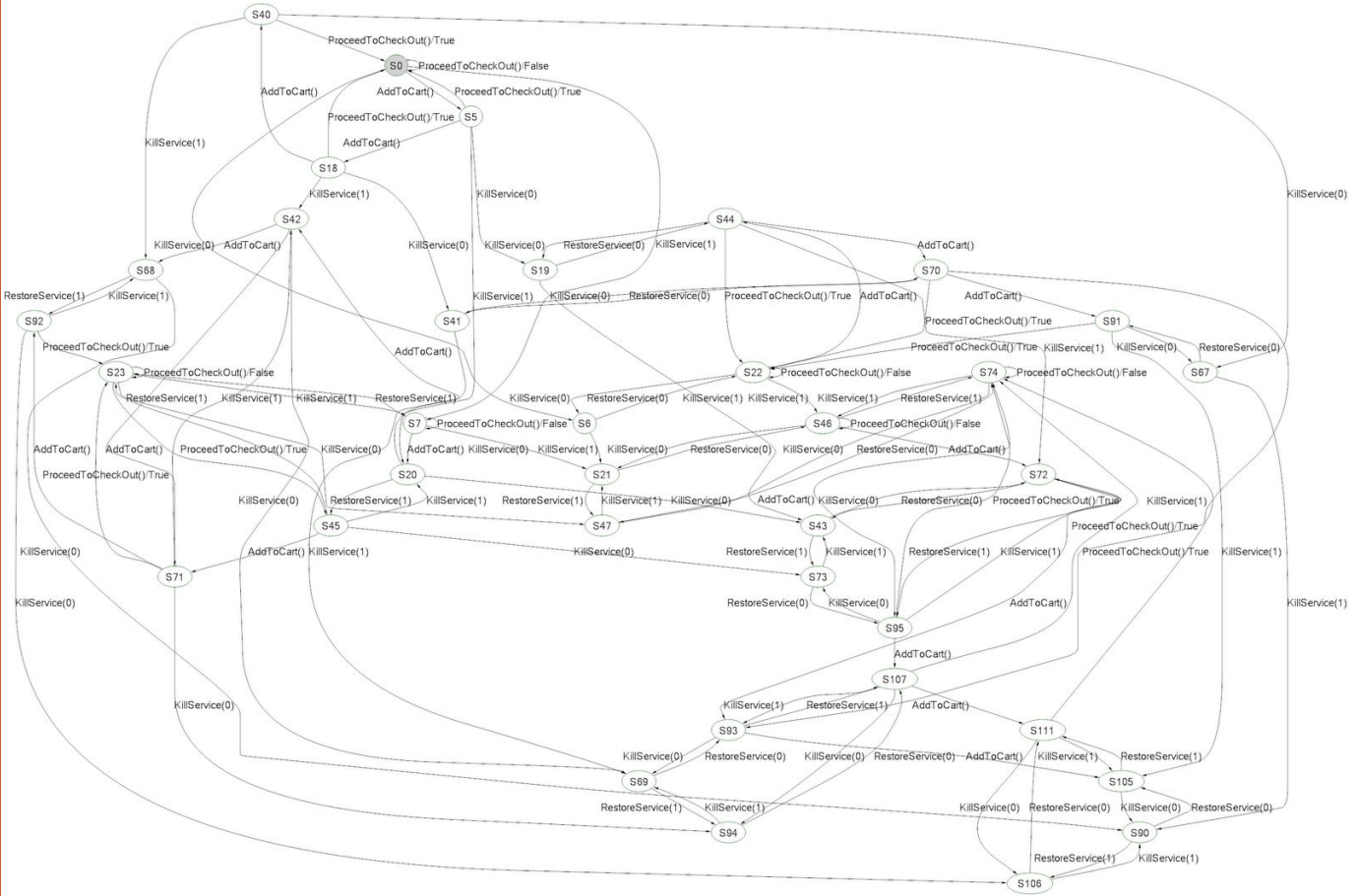
Examples


[New backlog item]
New
[Approved by the Product Owner]
[Reconsidered backlog item]
[Removed from the backlog]
Approved
Removed
[Removed from the backlog]
[Committed to by the team]
[Work stopped]
Committed
[Additional work found]
[Work finished]
Done

Examples


Create
Fix
Confirm
Fix
Verify
New
Confirmed
Fixed
Verified
Reopen
Reject
Reopen
Close
Reopen
Reject
Rejected
Closed

Examples


S40
S0
S5
S18
S42
S44
S68
S19
S70
S92
S41
S91
S23
S22
S74
S67
S7
S6
S46
S20
S21
S72
S45
S47
S43
S71
S73
S95
S107
S93
S111
S69
S105
S94
S90
S106
ProceedToCheckOut()/True
ProceedToCheckOut()/False
AddToCart()
KillService(1)
KillService(0)
RestoreService(1)
RestoreService(0)

Advantages/Disadvantages

Advantages

- Higher level of Automation is achieved
- Exhaustive testing is possible
- Changes to the model can be easily tested
- Easy test case/suite maintenance
- Reduction in Cost/Time savings
- Improved Test Coverage
- Can run different tests on n number of machines
- Early defect detection
- Improved tester job satisfaction

Disadvantages

- Requires a formal specification or model to carry out testing
- Changes to the model might result in a different set of tests altogether
- Test Cases are tightly coupled to the model
- Necessary Skills required in testers
- Bigger learning curve
- Sometimes it may be difficult to understand the model itself

Tools

Tools

	Conformiq Creator	DTM Tool
Description	**Uses a custom modeling language which is based on activity diagrams and a graphical domain specific action language**	**The DTM (Dialogues Testing Method) tool uses a custom activity model, and selects tests based on structural coverage**
Link	**https://www.conformiq.com/products/conformiq-creator/**	**http://www.dtmtool.com/**

Tools

	JSXM	MBTSuite
Description	SXM is model animation and test generation tool that uses a kind of EFSMs as its input. The generated tests can be transformed to JUnit test cases	Can generate test cases from UML models based on various coverage criteria or randomly
Link	http://www.jsxm.org/	http://www.mbtsuite.com/home.html

Tools

	ModelJUnit	MISTA
Description	Writes FSM or EFSM models as Java classes, then generate tests from those models and measure various model coverage metrics	MISTA generates test cases from high-level Petri nets, and using a mapping it can generate executable test code for various platforms
Link	https://sourceforge.net/projects/modeljunit/	http://cs.boisestate.edu/~dxu/research/MBT.html

Thank you for your time

https://github.com/Mosaal/TVVS