3rd UCAAT User Conference on Advanced Automated Testing

ETSI World Class Standards

Sophia Antipolis, French Riviera
20-22 October 2015

CUCUMBERSPEC: A TOOL THAT IMPROVES YOUR BEHAVIOUR-DRIVEN TESTING

by Rachid Kherrazi

NSPYRE

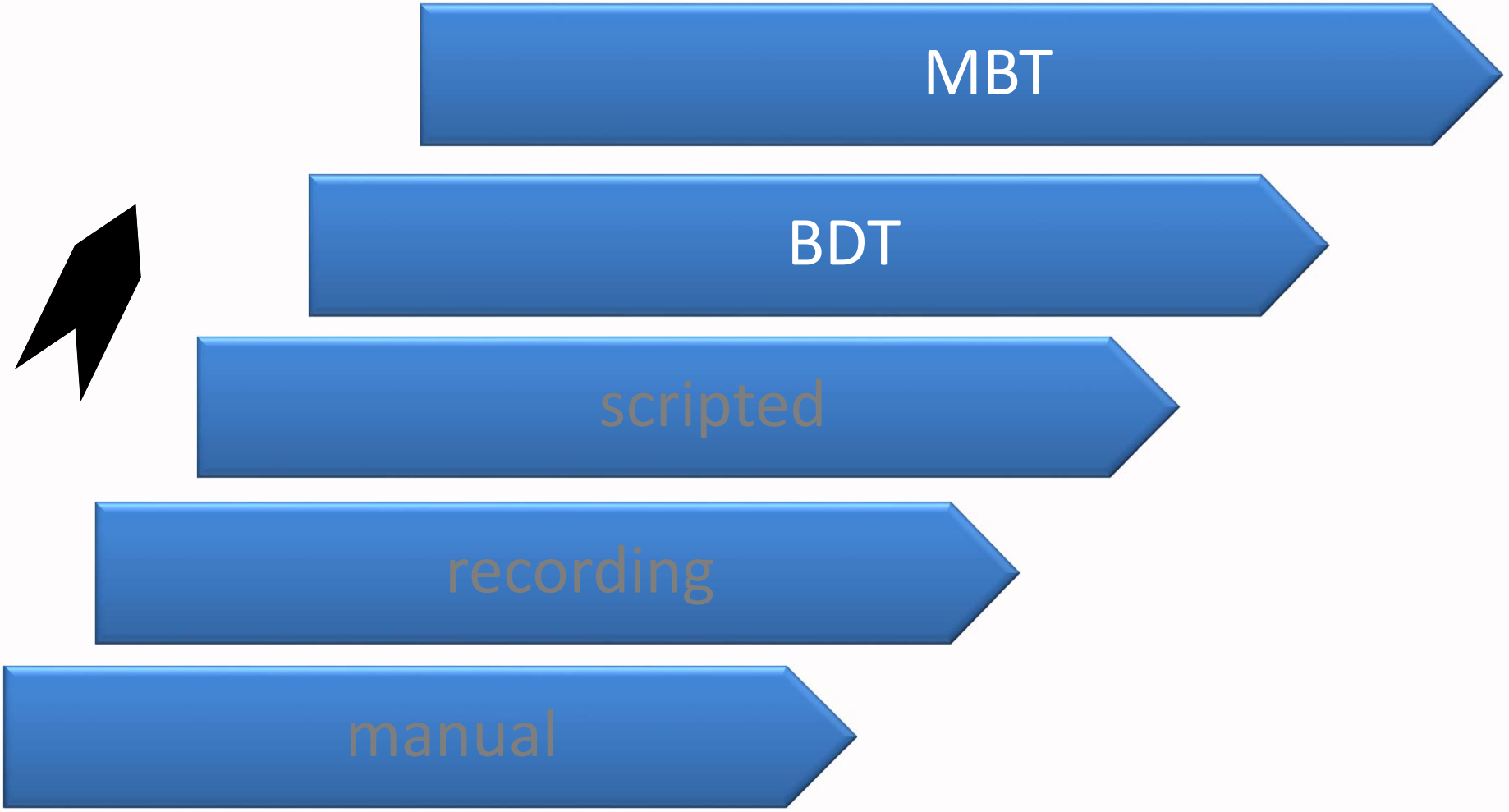
CONTENT

- Software testing evolution
  - BDT: Behavior Driven Testing
  - MBT: Model based testing
  - BDT + MBT
  - Benefits
- Case study and results

Rachid Kherrazi
Sr. Consultant @ Nspyre
Domain
Process and Product Improvement
Model Based Testing and Model Driven Engineering
iSQI Accredited Certified Model Based Tester Trainer

Also involved in this project
Kyra Hameleers
Adrian Yankov

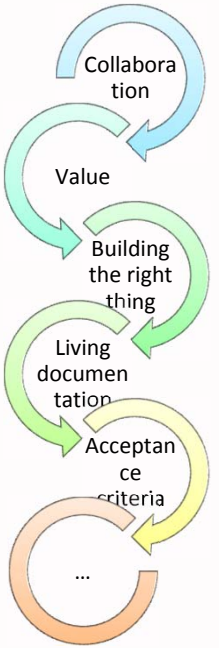
Software testing evolution



BDT – behavior driven testing

An approach to testing as an extension of BDD

- Specify the behavior of a system by looking at it from outside
- Domain driven
- Support collaboration
- Define a clear set of acceptance criteria
- Deliver stakeholder value
- Living documentation



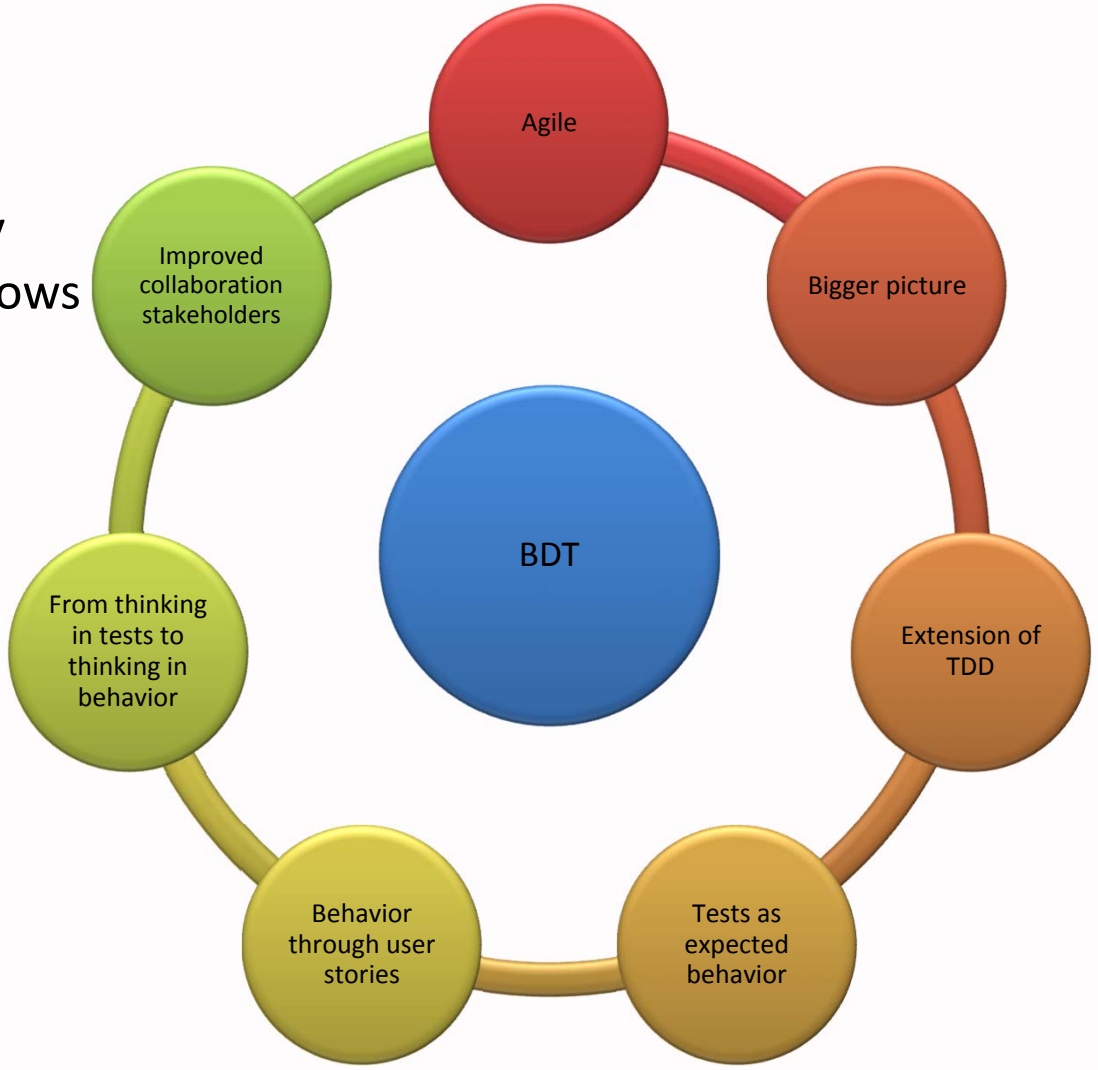
Given <an initial context>
When <event occurs>
Then <an outcome should be observed>

Scenario: Buy last coffee
Given there are 1 coffees left in the machine
And I have deposited 1 dollar
When I press the coffee button
Then I should be served a coffee

https://github.com/cucumber/cucumber/wiki/Feature-Introduction
https://en.wikipedia.org/wiki/Behavior-driven_development

BDT – a paradigm shift from functional to behavior

- User flows
- Understand new functionality
- Incrementally build up user flows
- Remove ambiguity
- Living documentation
- Automate what is important
- Test in sync
- Effective tests
- Onboarding
- Smart QA

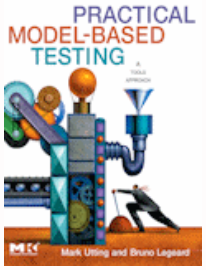


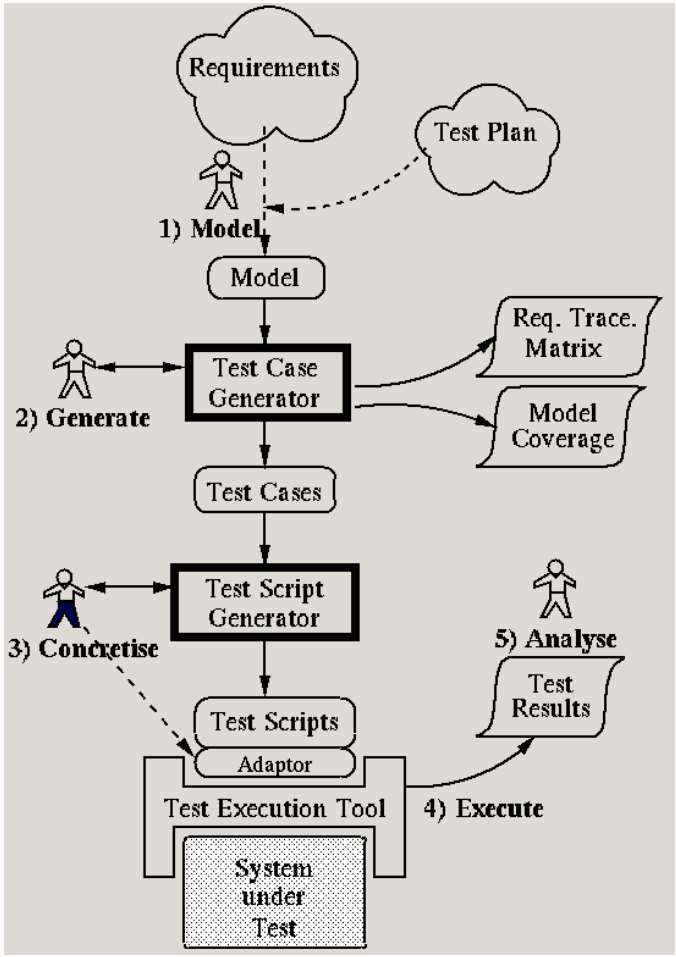
BDT- Behavior driven Testing

- Shortcomings
  - Manual design of test cases, mainly happy flow
  - Separate test scenarios, local test cases, limited interactions
  - Manual test data management, requires high effort to do more exhaustive testing, data permutation,…

Model based testing – an approach

1) create a **model** of the SUT
2) **generate** abstract test cases
3) make tests executable - **concretize**
4) **execute** tests on the SUT
5) **analyze** results





+ better communication using models
+ abstract tests
+ auto design of tests, variety of test suites
+ model checking
+ early exposure of ambiguities in spec/design
+ ease of updating of test suites
+ systematic coverage

o automatic execution
o auto regression testing

- modelling overhead
- complexity of the models

BDT + MBT

- Having advantages of both and Overcoming shortcomings
  - Generating Test models from test scenarios
  - Connecting different/isolated scenarios in test models
    - Creating a better overview of the behavior of the SUT(system under test)
    - Exposure of ambiguities in spec/design
    - Better analysis of the behavior of the SUT
    - Auto design/generation of test cases
    - More complex test cases
  - Test data management
    - Comprehensive test data management
    - Better test data coverage

Using Gherkin in MBT – novel idea

The novel **idea** behind using Gherkin in MBT:

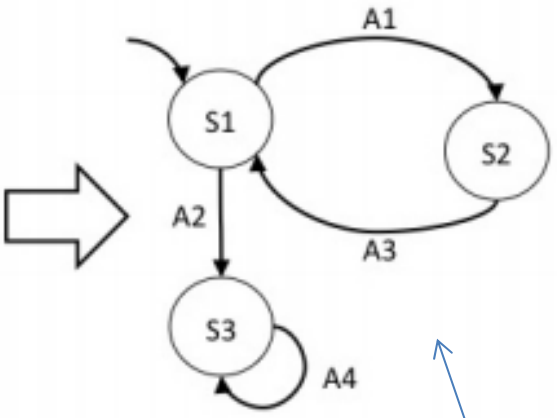
- Generate a *Test Model* from feature files, connecting the separate BDT scenarios in a model

Scenario: Test 1
Given I start on "S1"
When I "A1"
Then I should be on "S2"

Scenario: Test 2
Given I am on "S1"
When I "A2"
Then I should be on "S3"

Scenario: Test 3
Given I am on "S2"
When I "A3"
Then I should be on "S1"

Scenario: Test 4
Given I am on "S3"
When I "A4"
Then I should be on "S3"



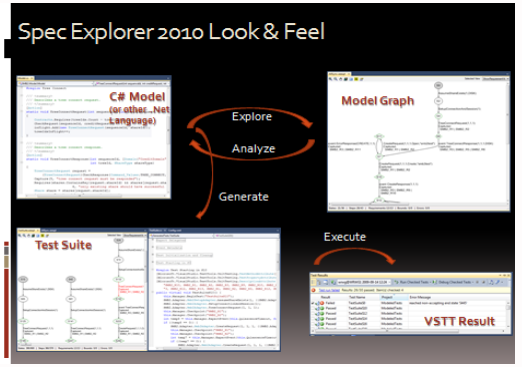
isolated test scenario's (in Gherkin)

test model (interaction between scenario's)

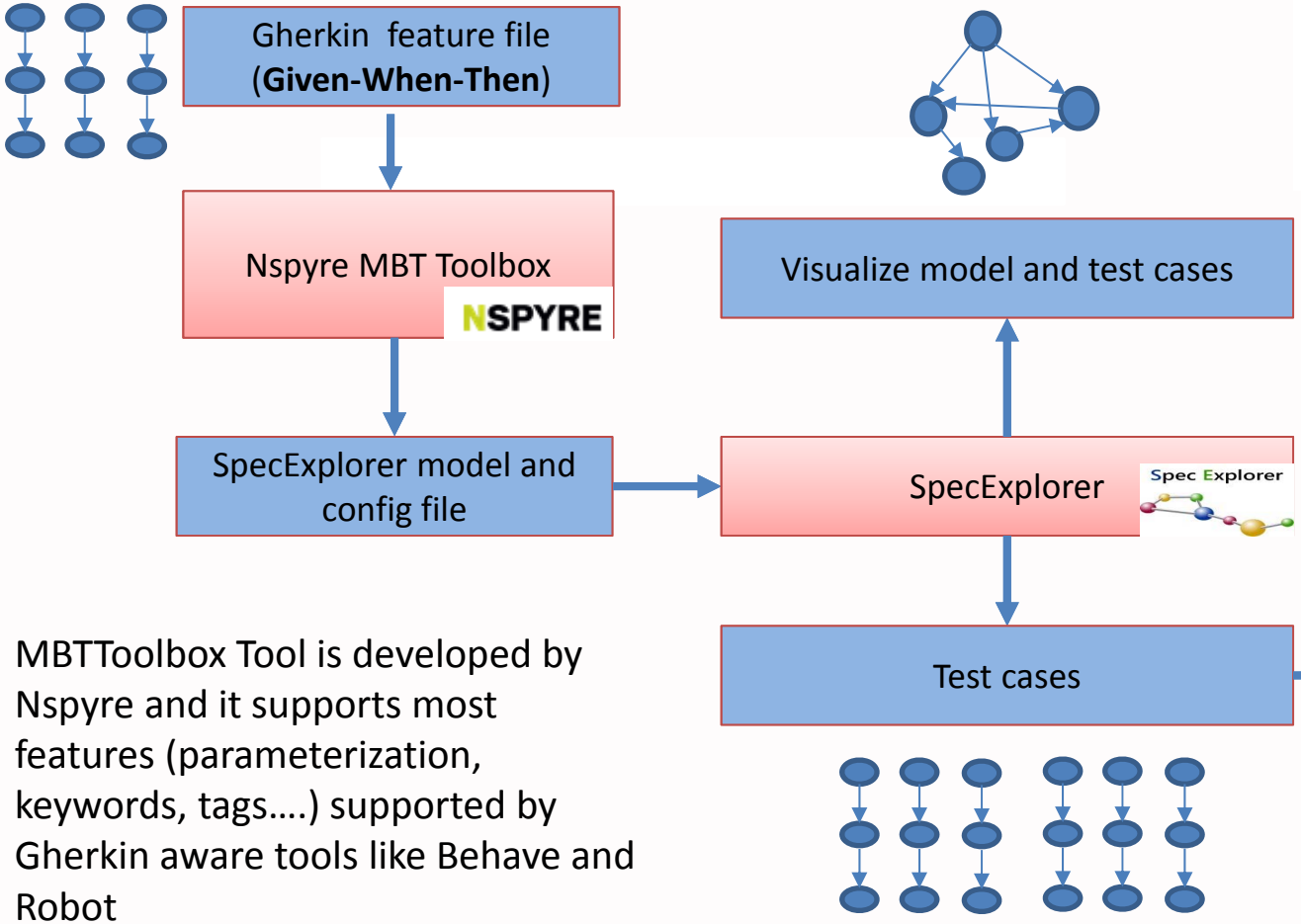
The MBT Toolbox generates a test model from the separate scenarios

First work by: Christian Colombo, Mark Micallef and Mark Scerri

Nspyre MBT toolbox



Winterop Team, Microsoft Corp.



MBTToolbox Tool is developed by Nspyre and it supports most features (parameterization, keywords, tags….) supported by Gherkin aware tools like Behave and Robot

CucumberSpec summary

Idea of our CucumberSpec

- Generates a Test model from a feature file, Connecting the separate scenarios in a model

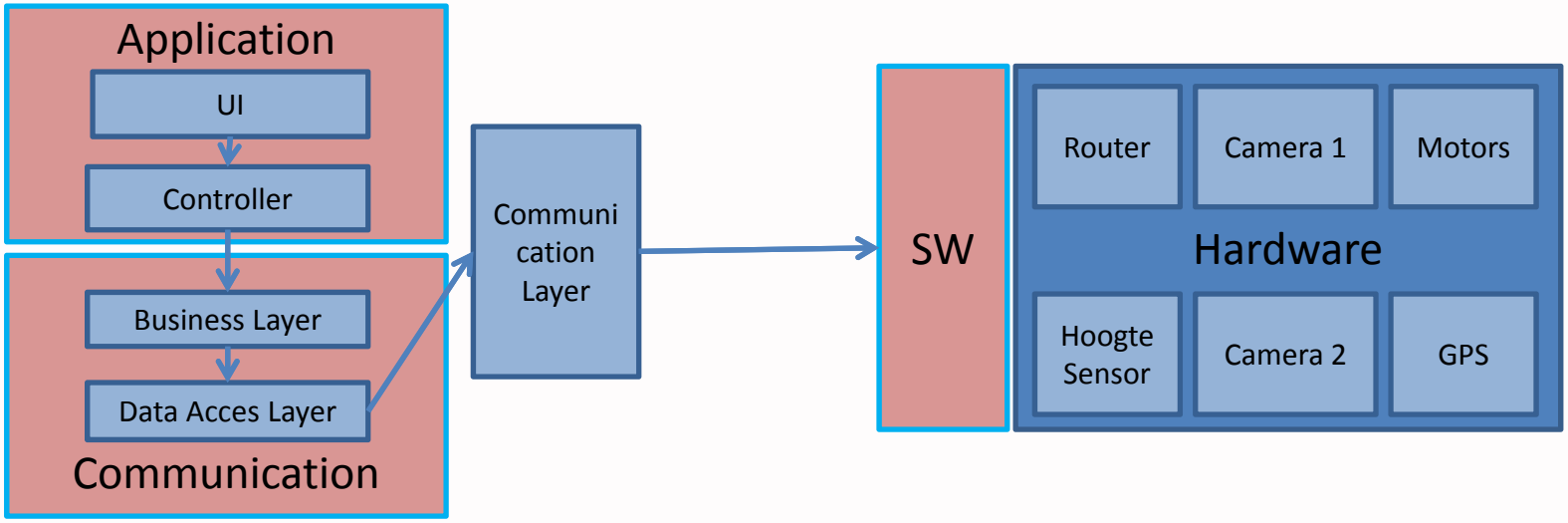
- Intention is to support most features (parameterization, tags….)supported by BDD tools like Behave and SpecFLow

- Get All Model Based Benefits,
  - Interaction between the scenarios
  - Generate more tests cases based on different coverage
  - No refinement is needed.
  - Fully automatic process.

Case Study: Ar.drone

- Mobile

AR.Drone



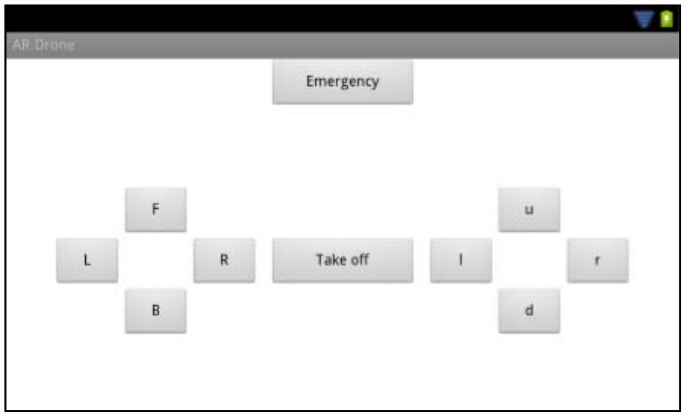
Test scope (Pilot: on limited requirements)

- Functional requirements
  1. Takeoff and Land via Android commands
     1. Takeoff
     2. Land
  2. Fly around via Android commands
     1. Left/Right
     2. Up/Down
     3. Forward/Backward
     4. Turn Lef/right
  3. Always be able to use the emergency command

Actions:

1. Take off
2. Emergency
3. Land
4. R: Right
5. L: Left
6. F: Forward
7. B: Backward
8. l: Turn left
9. r: Turn right
10. u: Up
11. d: Down

- States:
  1. Take off
  2. Flying
  3. Landing
  4. Landed
  5. Emergency



	Take off	Flying	Landing	Landed	Emergency
Take off	x	Reached 30 cm	Press land	X	Emergency
Flying	x	Right/ Left/ Up/ Down/ Forward/ Backward/ Turn left/ Turn right	Press land	x	Emergency
Landing	Press take off, height<0	Press take off, height>0	X	Reached 0 cm	Emergency
Landed	Press take off	X	X	x	Emergency
Emergency	x	Emergency, height>0	X	Emergency, height<0	x

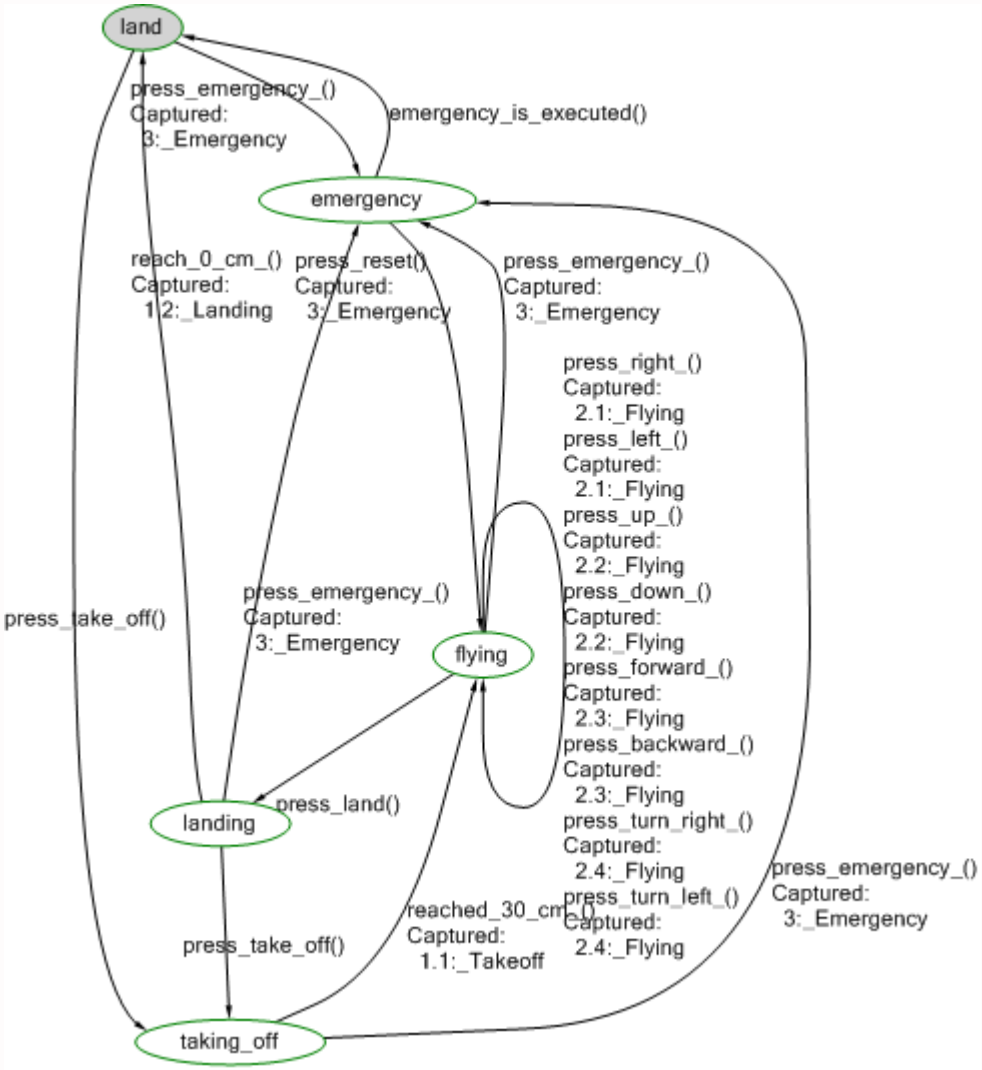
Created gherkin file

```
Feature: Flying a drone
@Taking off
Scenario: Taking off
Given I am on "land"
When I press take off
Then I am "taking off"
Scenario: Reached flying state
Given I am "taking off"
When I reached 30 cm REQ"1.1: Takeoff"
Then I am "flying"
Scenario: Taking off incorrect
Given I am on "landing"
When I press take off
Then I am "taking off"
@Landing
Scenario: Landing
Given I am "flying"
When I press land
Then I am "landing"
Scenario: Reached landed state
Given I am "landing"
When I reach 0 cm REQ"1.2: Landing"
Then I am "land"
Scenario: Landing incorrect
Given I am "take off"
When I press land
Then I am "landing"
```

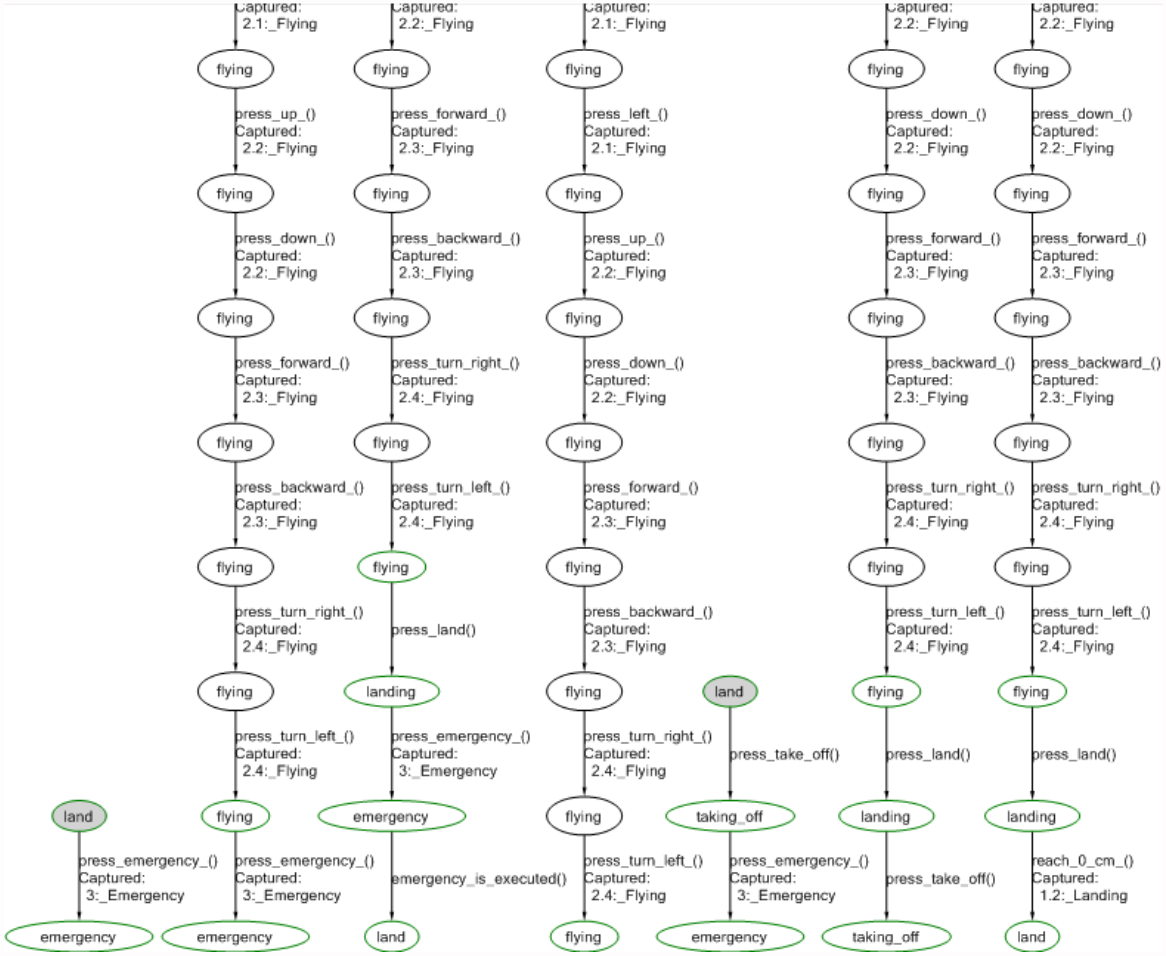
```
@Flying
Scenario: flying around
Given I am on "flying"
When I press right REQ"2.1: Flying"
Then I am "flying"
Scenario: flying around
Given I am on "flying"
When I press left REQ"2.1: Flying"
Then I am "flying"
Scenario: flying around
Given I am on "flying"
When I press up REQ"2.2: Flying"
Then I am "flying"
Scenario: flying around
Given I am on "flying"
When I press down REQ"2.2: Flying"
Then I am "flying"
Scenario: flying around
Given I am on "flying"
When I press forward REQ"2.3: Flying"
Then I am "flying"
Scenario: flying around
Given I am on "flying"
When I press backward REQ"2.3: Flying"
Then I am "flying"
Scenario: flying around
Given I am on "flying"
When I press turn right REQ"2.4: Flying"
Then I am "flying"
Scenario: flying around
Given I am on "flying"
When I press turn left REQ"2.4: Flying"
Then I am "flying"
```

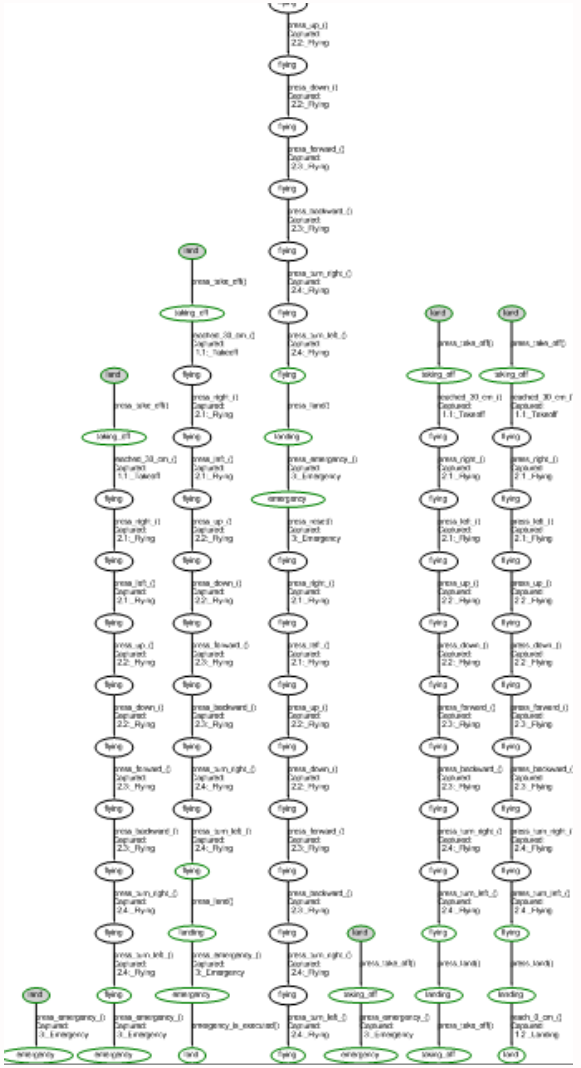
```
@Emergency
#Emergency means the drone will turn all motors of
Scenario: Emergency
Given I am on "flying"
When I press emergency REQ"3: Emergency"
Then I will be in "emergency"
Scenario: Emergency
Given I am "landing"
When I press emergency REQ"3: Emergency"
Then I will be in "emergency"
Scenario: Emergency
Given I am "land"
When I press emergency REQ"3: Emergency"
Then I will be in "emergency"
Scenario: Emergency
Given I am on "taking off"
When I press emergency REQ"3: Emergency"
Then I will be in "emergency"
#If emergency is over before the crash
Scenario: Recover Emergency
Given I am on "emergency"
When I press reset
Then I am on "flying"
#after crash
Scenario: Full emergency
Given I am in "emergency"
When emergency is executed
Then I am "land"
```

GENERATED MODEL

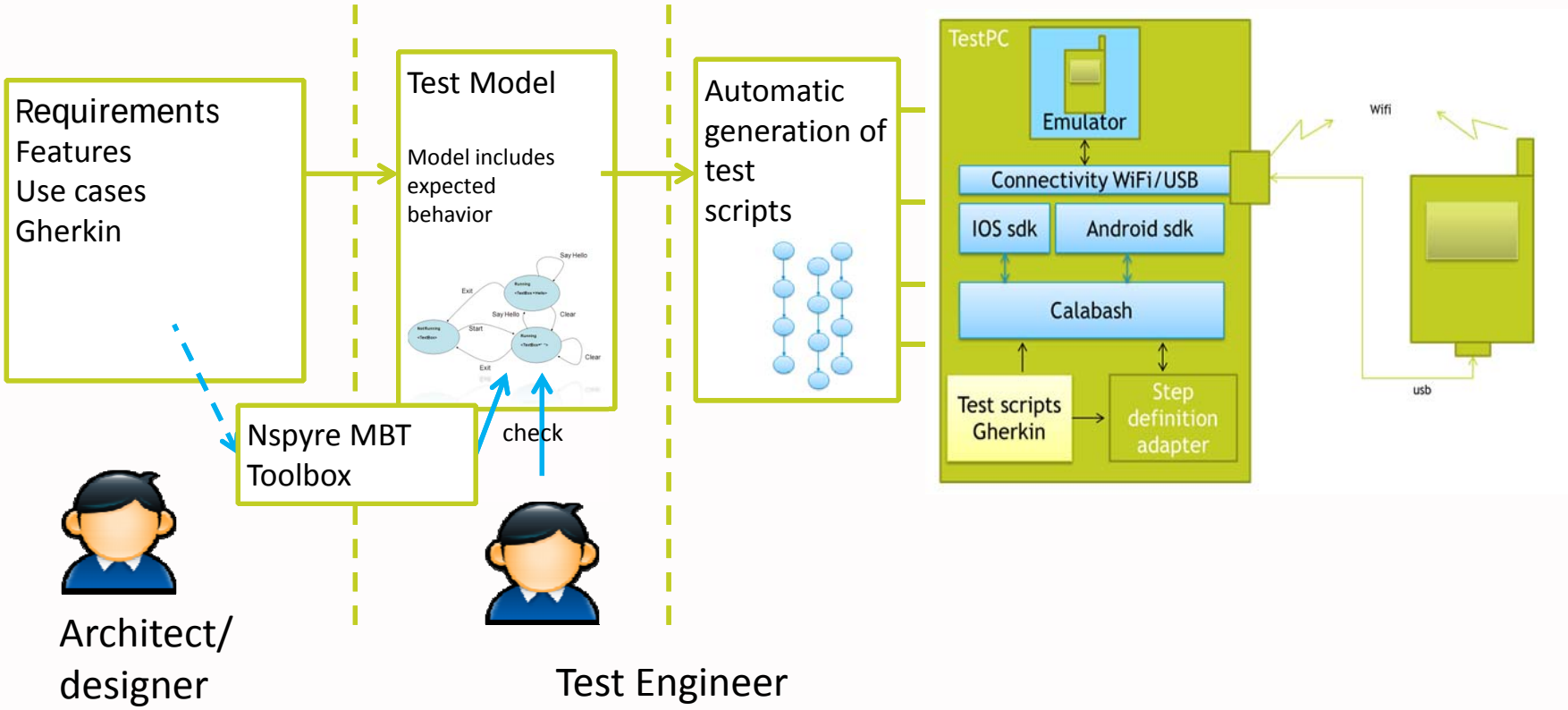


Test cases





Workflow



Model Based Crossplatform Mobile testing framwork

CusumberSpec Tool

Current version of tool supports:

- Parameterization
- Requirements traceability
- Support of other MBT tools (NModel, PyModel) and export of test cases to other scripting (Python, C#).
- Possibility to automate step definition, if keywords or call function are used.

- Next step: applying approach in a high-tech industry case through a joint effort

Questions?

MBT is…..

Strategies, tools and artifacts

Manage complexity

Reduces the need of manual or human involvement of interaction

Avoids spending time in unskilled repetitive error prone or redundant tasks

Provides bandwidth to Innovate!!

results

Metrices	Cucumber	MBT	CucumberSpec
Approach	using gherkin files and action definition to generate and execute test suite	using models to generate and execute test suite	using gherkin files to generate models, and generate and execute test suite
Technique	Behavior Driven Development	Model Based Testing	Combining strength of BDD and MBT
Modeling	Create gherkin files	Create test model	Generate test model
Effort	High	Medium/High	Medium
Cost(hr)	8	6	4
Number of testcases	20, 0-switch	7, chinese postman	7, chinese postman