ANR AAPG 2018 – PHILAE Project

Project Presentation

Pr. Bruno Legeard – Scientific Coordinator
Pr. Roland Groz – project leader for LIG (Grenoble)

From Model-Based Testing to Cognitive Test Automation

PHILAE – Mission Statement

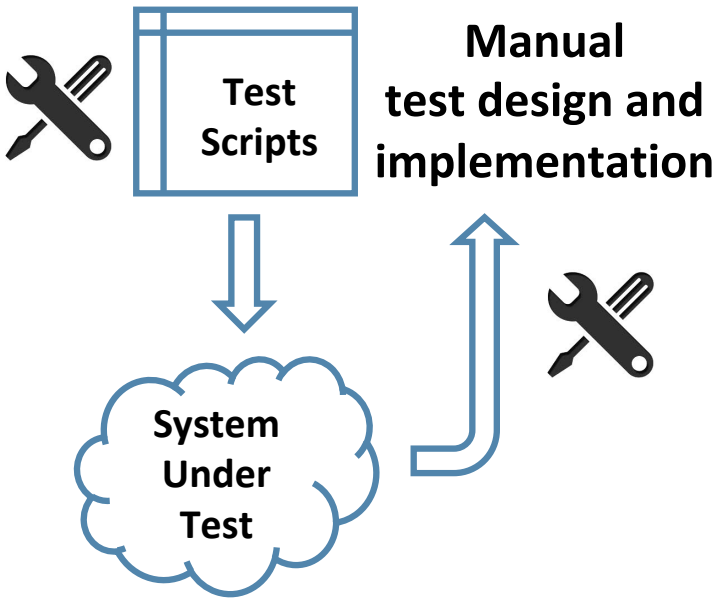
"PHILAE aims to fully automate the creation and maintenance of automated regression tests based on system execution traces (in production and testing) and other software lifecycle metadata in the context of iterative-incremental software development"

PHILAE Team

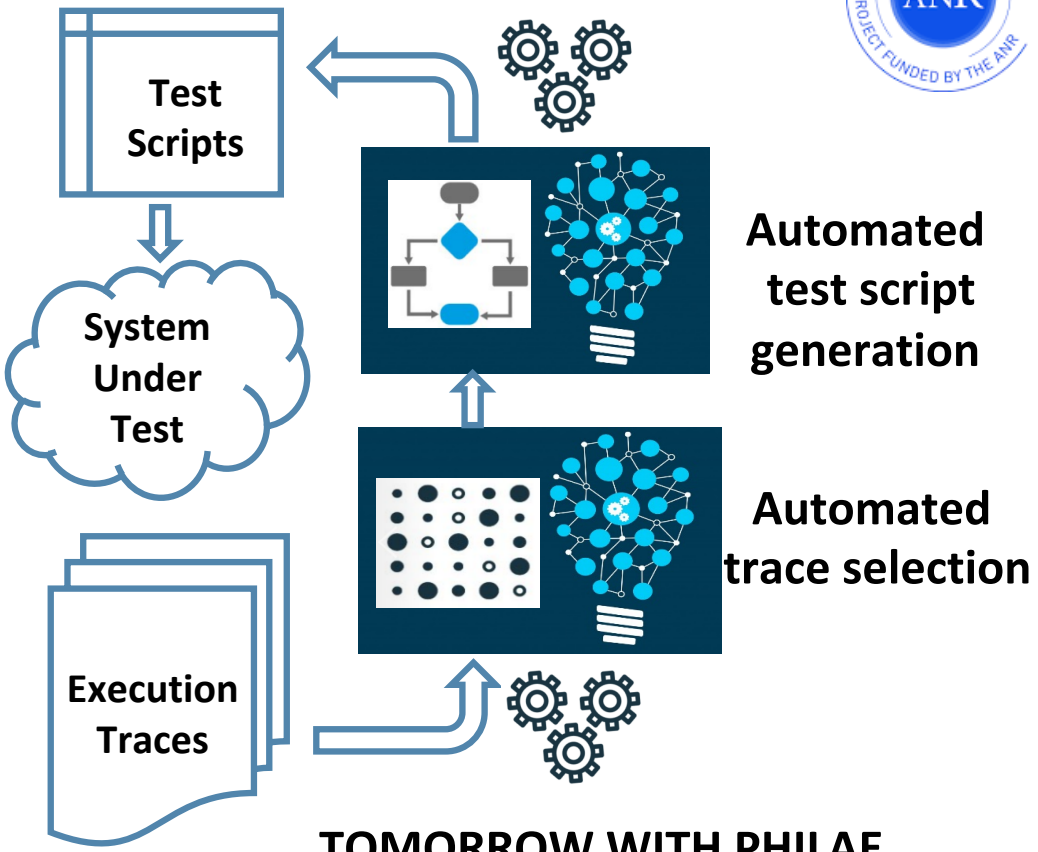
- UBFC
  - Bruno Legeard
  - Frédéric Dadeau
  - Fabrice Bouquet
  - Vahana Dorcis (PhD student)
  - Antoine Chevrot (PhD student)
- Grenoble INP - LIG
  - Roland Groz
  - Christophe Brouard
  - Yves Ledru
  - Lydie du Bousquet
  - Catherine Oriat
  - German Vega (Eng)
  - Nicolas Bremond (Eng)
  - William Ochoa (PhD student)
- Orange Labs Services
  - Yann Helleboid (Lannion)
  - Benoît Parreaux (Lannion)
  - Yannick Dubucq (Bordeaux)
  - Edgar Fernandes (Bordeaux)
  - Pierre Nicoletta (Paris)
- Smartesting (Besançon)
  - Elizabeta Fourneret
  - Julien Botella
- Univ. Sunshine Coast
  - Mark Utting
- Simula Research Lab
  - Arnaud Gotlieb

Issue: S/W dvt bottleneck

- Fact 1: S/W testing is becoming a bottleneck
  - Continuous integration of large code bases (e.g. Google, Salesforce, DS…)
  - Large and ever expanding regression test suites
  - Overnight runs (may climb > 24h)
  - Average level of automation for test activities: 16% (World Qual. Rep. 2018)
- Fact 2: Model-Based Testing improves quality but not cost effectiveness
  - Complexity, deployment
  - 14% penetration level (of test professionals, Techwell 2017)

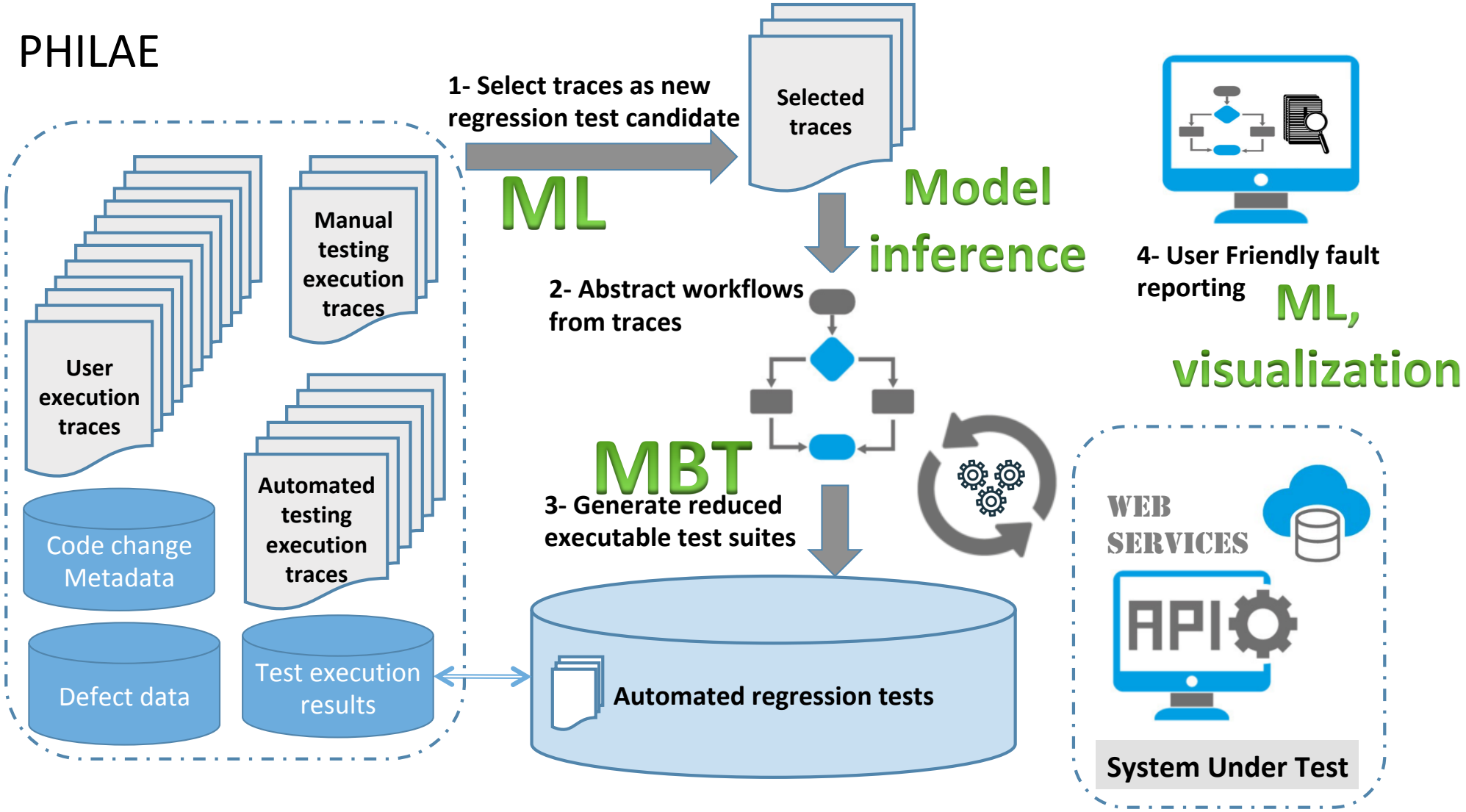


TODAY



TOMORROW WITH PHILAE

PROJET FINANCÉ PAR L'ANR
ANR
PROJECT FUNDED BY THE ANR

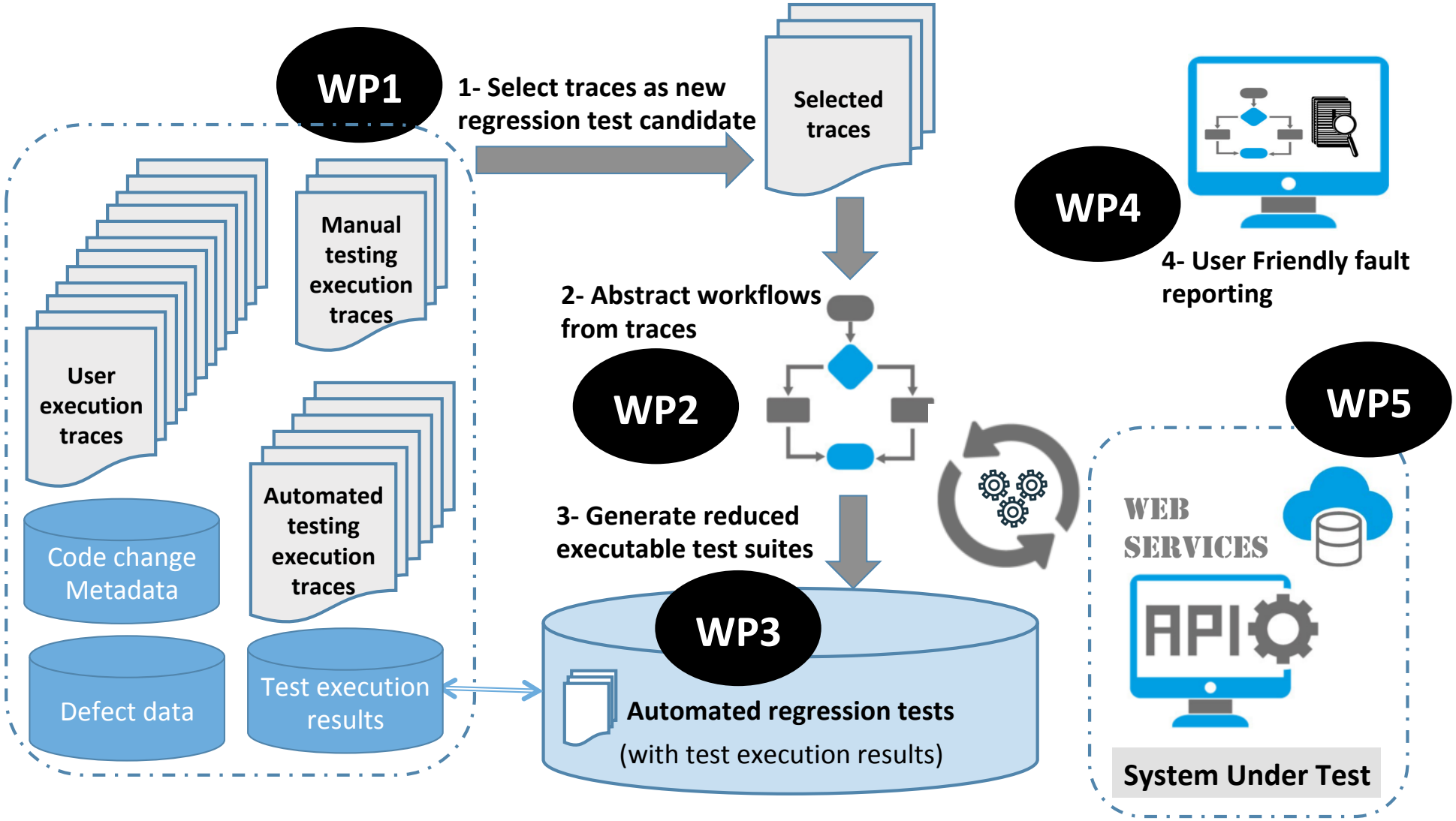

PHILAE
1- Select traces as new regression test candidate
ML
Selected traces
Model inference
4- User Friendly fault reporting
ML, visualization
2- Abstract workflows from traces
MBT
3- Generate reduced executable test suites
Manual testing execution traces
User execution traces
Automated testing execution traces
Code change Metadata
Defect data
Test execution results
Automated regression tests
WEB SERVICES
API
System Under Test

PHILAE – Identified Research challenges

- Selecting test and operational execution traces to satisfy “good enough” coverage for the automated regression tests
- Producing automated executable tests from selected execution traces
- Prioritizing dynamically automated regression automated test scripts and minimizing the whole generated regression test suite
- Producing an explanation and visualization of the coverage of what is automatically produced

PHILAE – Identified Research Directions

- Selecting test and operational execution traces
  - ✓Clustering traces → see SMA work on legacy test case analysis and refactoring
  - ✓Model learning from traces → from LIG Background
- Producing automated executable tests from selected execution traces
  - ✓ Learning automated test actions
  - ✓ Mapping traces to sequences of test actions
- Prioritizing dynamically automated regression automated test scripts and minimizing the whole generated regression test suite
  - ✓ Define this as a constraint optimization problem and solve it → from SRL Background
- Producing an explanation and visualization of the coverage of what is automatically produced
  - ✓ Coverage analysis from learned models → From SMA background


WP1
1- Select traces as new regression test candidate
Selected traces
WP4
4- User Friendly fault reporting
Manual testing execution traces
User execution traces
2- Abstract workflows from traces
WP2
WP5
Automated testing execution traces
Code change Metadata
3- Generate reduced executable test suites
WEB SERVICES
API
WP3
Defect data
Test execution results
Automated regression tests
(with test execution results)
System Under Test

PHILAE case studies

- Orange Labs Services – Live Box
- USC – Schoolbus (mobile and web app)
- Smartesting – (with Flexio) Industrial processes
- FEMTO-ST – Scannette (e-cart supermarket) + medical imaging s/w

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

orange™

Grenoble INP

Smartesting®
Solutions & Services

simula

USC
AUSTRALIA

Livebox case study

- Final checks before deployment of firmware
- Soak testing (endurance) : simulating user actions over weeks without reboot
- Hundreds of GB of test traces (No user trace – privacy issues)
  - Black box traces (I/O observed from test harness)
  - Perf monitoring of Livebox (cpu, mem, disk, netw, wifi…) – separate traces

Goals

- Enhance variety of tests, in constant test budget
- Automate bug analysis/detection (several issues/day -> few new bugs/month)

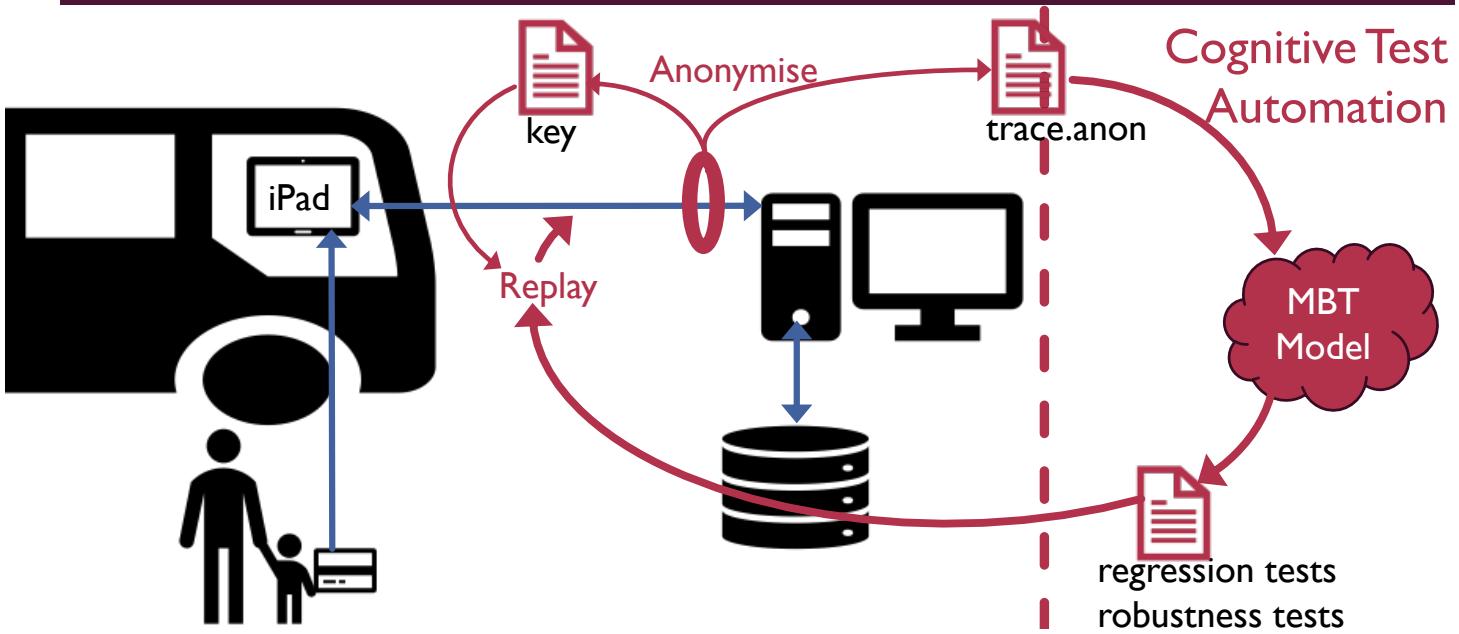
Livebox case study challenges

- Endurance vs functional testing
  - Potentially very long time between cause (defect) and detection
  - Very different from usual MBT
- No user trace, no workflow (repetitive service invocations)
- Traces already very high-level (no need to abstract from low-level API)
- Distributed testing (with interleaving)

Currently Investigating:

- Recognizing anomaly patterns (from monitoring+exec traces)
- Detecting weak signals (potential causes)

SCHOOL BUS SYSTEM ARCHITECTURE



Schoolbus traces

Features

- Server records students on each bus run
- Students swipe an ID card as they enter or leave
- Parent notified (SMs or e-mail) when child gets off
- Server tracks progress of bus (with GPS from bus)

- All events -> XML record

One student checks into the bus by swiping their card


```
<?xml version='1.0' encoding='UTF-8'?>
<RequestWrapperOfStatusOutput>
  <Time>2018-09-14T07:43:16.7749833+10:00</Time>
  <Origin>BUS23</Origin>
  <Path>/webservice/SchoolMobileWS.asmx/SNSCheckIn</Path>
  <Request>username=USER417&amp;amp;password=PASS949&amp;amp;studentID=1595&amp;amp;run=RUN364&amp;amp;time=2018-09-14T07:43:04.213&amp;amp;latitude=???&amp;amp;longitude=???</Request>
  <Response>
    <Status>0</Status>
    <ClientCode>GAT</ClientCode>
  </Response>
</RequestWrapperOfStatusOutput>
```

HOW TO TEST? VERSION 1: SMART TESTER

1. Analyse test traces
   - Manual Inspection, Python (Jupyter Notebook, visualisation, etc)
2. Choose some typical traces
   - LPMAAA.....A.....i...i...i...i..i...A.....i....i.....i.....iiO....
   - LPMA.....I.....o.....o..o.....o..o....o..ooC....o.....
3. Design an MBT model
   - Simple FSM plus set of students
4. Generate some (1) Regression Tests; (2) Robustness Tests
   - (1) just replay; (2) add bad inputs, bad transitions, etc.
5. Replay on SUT

Current status (1st semester)

- Identifying case studies + issues and data
- Data analysis and preparation
- Clustering to identify typical traces and behaviours

Next

- Better clustering − classification (association with failures)
- Trace selection
- Reification (rebuilding tests from abstract traces patterns)