Automation Supported Testing of Automation Systems based on Test-First Development

Dietmar Winkler  Stefan Biffl
Christian Doppler Laboratory “Software Engineering Integration for Flexible Automation Systems” (CDL-Flex)
Institute of Software Technology and Interactive Systems (ISIS)
Vienna University of Technology
http://cdl.ifs.tuwien.ac.at
Context & Motivation

Software components in automations systems
- Added value provided by software components (software-intensive systems).
- Realization of functional behavior in software components.
- Increased flexibility (e.g., response to changing requirements, reconfiguration).
- Delivery of (tested) releases within short iterations.

Challenges and Goals
- Functional, testing, and diagnosis aspects are scattered over the code and hinder efficient automation systems testing.
  → Need for efficient testing methods and automated testing strategies.
- Limitations in systematic development processes.
  → Need for flexible and systematic systems development processes.

Application of Best-Practices derived from business IT software development
- Test-first (test-driven) development.
- Continuous Integration and systematic testing.
- Automation-supported test case generation, execution, and reporting.
- Prototype application “Bottle Sorting Application” for evaluation purposes.
Test-Driven Development Steps:
1. **Think**: (a) selection of new requirements and (b) test case definition.
2. **Red**: Implementation and execution of test cases (failed).
3. **Green**: Implementation of functionality and test case execution until all tests are successful.

**Continuous Integration and Test:**
- Frequent test runs
- Immediate Feedback on test results (e.g., daily builds)
- Efficient regression testing.
- Automation and tool support

---

Foundation for Automation Supported Testing
Model-Driven Testing based on UML

Basic Test Levels in Automation Systems

Systematic Development Processes, e.g., based on the V-Modell XT*, enable automation supported testing on various levels.

- **System Test Level** based on requirements and use cases.
- **Integration Test Level** based on architecture, components, and the interaction between components.
- **Unit Test Level** based on individual components.

*V-Modell XT: http://www.v-modell-xt.de
Research Approach

Need
- Automation supported and flexible systems development processes and efficient testing in automation systems projects.

Approach
- Development of an **automated testing framework** that supports frequent test runs.
- Definition of a **test management process** approach
- First evaluation in a **prototype application**: bottle sorting application.
Test Framework for Test-First Development of Automation Systems

1. Test case generation
2. Test case execution on target environment
3. Capture test and diagnosis results
4. Analyzing test results
5. Test reporting
Test Process

- **Step 1: Automation supported test case generation**
  a) Capturing basic systems requirements.
  b) Test Scenarios based on Use Cases.
  c) Automation supported test case generation.
  d) Test-Framework with keyword driven test.

- **Step 2: Test case execution on target system**
  a) Upload code to target system (modeled in function blocks).
  b) Logging of diagnosis data.

- **Step 3: Capture test and diagnosis results**
  - Capture results assigned to test cases and test scenarios.

- **Step 4: Analyzing test results**
  - Analyzing test results.

- **Step 5: Test reporting**
  - Generate test report, e.g., following the continuous integration and test strategy.
Sorting Application Prototype: Capturing Basic Systems Requirements

- Bottle sorting application
  - Identification of individual bottles on a conveyor (stopper unit).
  - Move identified bottles to the second conveyor (handling unit).
  - Stop at the appropriate loading station (sorting unit).
  - Move sorted bottles to an appropriate box (according to the colour).
- Focus on the handling unit.
Expected user behaviour on requirements level from user perspective.
Use cases drive the composition of test scenarios.
Handling unit picks one bottle from conveyor 1 to conveyor 2.

Advantages:
- Common “language” between different disciplines.
- Enhanced understanding of the customer requirements.
- Test scenarios as vehicle for communication between stakeholders.

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>Scope</th>
<th>Pre-condition</th>
<th>Action</th>
<th>Expected Result / Post-Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sorting a Bottle</td>
<td>System: Bottle sorting application</td>
<td>Handling Unit in idle position</td>
<td>Command to sort bottle</td>
<td>Handling unit in idle position and part sorted</td>
</tr>
<tr>
<td>1.1</td>
<td>Recognizing Bottle at Conveyor 1</td>
<td>Subsystem: Stopper</td>
<td>No bottle present</td>
<td>Stopping bottle</td>
<td>Bottle stopped by stopper unit</td>
</tr>
<tr>
<td>1.2</td>
<td>Moving Bottle from Conveyor 1 to 2</td>
<td>Subsystem: Handling Unit</td>
<td>Bottle available Bottle available</td>
<td>Gripping, moving, and releasing bottle</td>
<td>Bottle moved to conveyor 2 Gripper returned to idle position</td>
</tr>
</tbody>
</table>
Sorting Application Prototype: Automation Supported Test Case Generation

- Behaviour diagram (state chart) as foundation for automated test case generation.
- Test cases can be derived directly from state charts.
- State charts should cover all states and the overall specification.
- Test Scenario “Sorting a Bottle”
  - Subsystem “Handling Unit”: Moving bottle from Conveyor 1 to Conveyor 2 (components: stopper unit & handling unit).
  - Subsystem “Sorting Unit”: Stopping and sorting bottle to appropriate box.

1.2.1

Subsystem “Handling Unit”: Moving bottle from Conveyor 1 to Conveyor 2 (components: stopper unit & handling unit).

Subsystem “Sorting Unit”: Stopping and sorting bottle to appropriate box.
Sorting Application Prototype: Derived Test Cases from State Chart

- Automation supported test case generation based on transitions of the state charts.
- Definition of test scenarios (sequences of individual test cases).

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>Pre-condition</th>
<th>Action</th>
<th>Expected Result / Post-Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>Stopper Unit: Stop bottle</td>
<td>State=Stopper.Contracted</td>
<td>Bottle arrived</td>
<td>Stopper extended &amp; bottle stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No bottle present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1.2</td>
<td>Stopper Unit: Bottle removed</td>
<td>State=Stopper. Extended Bottle present</td>
<td>Bottle removed</td>
<td>Stopper contracted</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Handling Unit: Grip bottle</td>
<td>State=(Stopper.Extended &amp;&amp; HandlingUnit.Ungrip &amp;&amp; HandlingUnit.Left)</td>
<td>Bottle arrived</td>
<td>State=(Stopper.Extended &amp;&amp; HandlingUnit.Grip)</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Handling Unit: Release bottle</td>
<td>State=HandlingUnit.Right &amp;&amp; HandlingUnit.Grip</td>
<td>Ungrip</td>
<td>State=HandlingUnit.Right &amp;&amp; HandlingUnit.Ungrip</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Handling Unit: Move and release bottle</td>
<td>State=(Stopper.Extended &amp;&amp; HandlingUnit.Left)</td>
<td>Move to Right</td>
<td>State=(Stopper.Extended &amp;&amp; HandlingUnit.Grip &amp;&amp; HandlingUnit.Right)</td>
</tr>
<tr>
<td>1.2.4</td>
<td>Handling Unit: Return to idle state</td>
<td>State=HandlingUnit.Right</td>
<td>Wait 8ms &amp;&amp; Ungrip</td>
<td>State=HandlingUnit.Left</td>
</tr>
</tbody>
</table>
Sorting Application Prototype:
Keyword Driven Test: Definition & Execution

- Keyword-Driven Test based on a Spreadsheet (Excel)
- Transform and execute (selected) tests on a target platform, e.g., applying logi.CAD/RTS.

![Diagram](image1)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Resource</td>
<td>Offline-Conf/Offline-Tests</td>
</tr>
<tr>
<td>2</td>
<td>Type of Test</td>
<td>OFFLINE</td>
</tr>
<tr>
<td>3</td>
<td>System / Component</td>
<td>Test Id</td>
</tr>
<tr>
<td>4</td>
<td>Stopper Unit</td>
<td>T1.1</td>
</tr>
<tr>
<td>5</td>
<td>Stopper Unit</td>
<td>T1.1</td>
</tr>
<tr>
<td>6</td>
<td>Stopper Unit</td>
<td>T1.1</td>
</tr>
</tbody>
</table>
| 7 | Handle Unit | T1.2 | Moving bottle from Conveyor 1 to State 1 (Stoper.pdf, Extended 
| & HandlingUnit, Ungrip & HandlingUnit, Left) |
| 8 | Ergebnis | Zyklus/Zeit | Aktion |
| 9 | WAHR | 0 START | ACTIVATE | FALSE |
| 10 | WAHR | 1 FORCE | A | FALSE |
| 11 | WAHR | 1 FORCE | B | FALSE |
| 12 | WAHR | 1 FORCE | TimeDiscrepanz | 1000 |
| 13 | WAHR | 1 WAIT | READY | FALSE |
| 14 | WAHR | 1 WAIT | DiagCode | 0x8000 |
| 15 | WAHR | 1 WAIT | FORCE | TRUE |
| 16 | WAHR | 1 WAIT | SF_EquivalentOut | TRUE |
| 17 | WAHR | 1 WAIT | Error | FALSE |
| 18 | WAHR | 1 WAIT | DiagCode | 0x8000 |
| 19 | WAHR | 5 STOP | Anweisungen vor dieser Zeile einfügen | Anweisungen vor dieser Zeile einfügen |

http://www.logidiag.at/id/logi.DIAG/
Sorting Application Prototype: Capture test and diagnosis results

Keyword driven test – Execution Steps:
- Upload statements and Test Parameters to target system.
- Sequential execution of individual statements.
- Logging of diagnosis data for defect detection & traceability.
- Download test case results to Host-System

Sample Test Case Execution on Target System with logi.CAD/RTS
Sorting Application Prototype: Analyzing Test Results

- Individual test cases are based on transitions (change of states).
- Test scenarios encapsulate a defined set of test cases (metric: test coverage)
- Requirements include a set of test scenarios.
- Aggregation of Statement/Test case results on scenario and/or requirements level.
Sorting Application Prototype: Test reporting

- Aggregation of test results and logging data.
- Project management.
  - Project Progress in terms of completed software functions.
  - Quality Status: test case results and test coverage.

![Diagram showing requirements and test case execution](image-url)
Lessons Learned & Future Work

- Increased flexibility and (software) complexity in the automation systems domain lead to new challenges in software construction.

- Lessons learned from business IT software development can help systems engineers in constructing high-quality products in short iterations.

- Lessons learned from a pilot application showed the expected benefits in a small show case application.
  - Systematic engineering process support based on the V-Modell XT and Test-First Development (TFD) on various levels.
  - (Automated) test case generation lead to frequent test runs and continuous engineering project monitoring and control.

- Future work includes
  - Refining the process model and the keyword driven test approach.
  - Investigating the scalability of the test framework in a larger project context.
  - Elaborating on a larger pilot application with industry partners with focus on data collection to empirically investigate the expected benefits.
Backup
State charts are common practices in the automation systems domain.

- Ability for automated code generation.
- Modelling of state charts including error states.
- Example: handling unit on component level.

### Sorting Application Prototype: Integration and Unit Tests

<table>
<thead>
<tr>
<th>No</th>
<th>Desc.</th>
<th>Level</th>
<th>Type*</th>
<th>Pre-condition</th>
<th>Input</th>
<th>Expected Result</th>
<th>Post-condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gripper move to Pos</td>
<td>Comp.</td>
<td>NC</td>
<td>Handling Unit idle</td>
<td>Sort part</td>
<td>Gripper moved to intended position</td>
<td>Gripper is in intended position</td>
</tr>
<tr>
<td>2</td>
<td>Axis got stuck</td>
<td>Comp</td>
<td>EC</td>
<td>Handling Unit in idle Position</td>
<td>Sort part, error after 3s</td>
<td>Positioning Unit reports an error; Handling Unit idle</td>
<td>Handling Unit in idle position</td>
</tr>
</tbody>
</table>