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

logi.cals®

OESTERREICHISCHE COMPUTER GESELLSCHAFT
AUSTRIAN COMPUTER SOCIETY

TU VIENNA

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.**

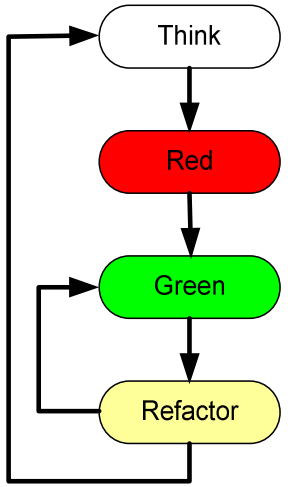
Application of Best-Practices derived from business IT software development

- 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.

Foundation for Automation Supported Testing
Test-First Development

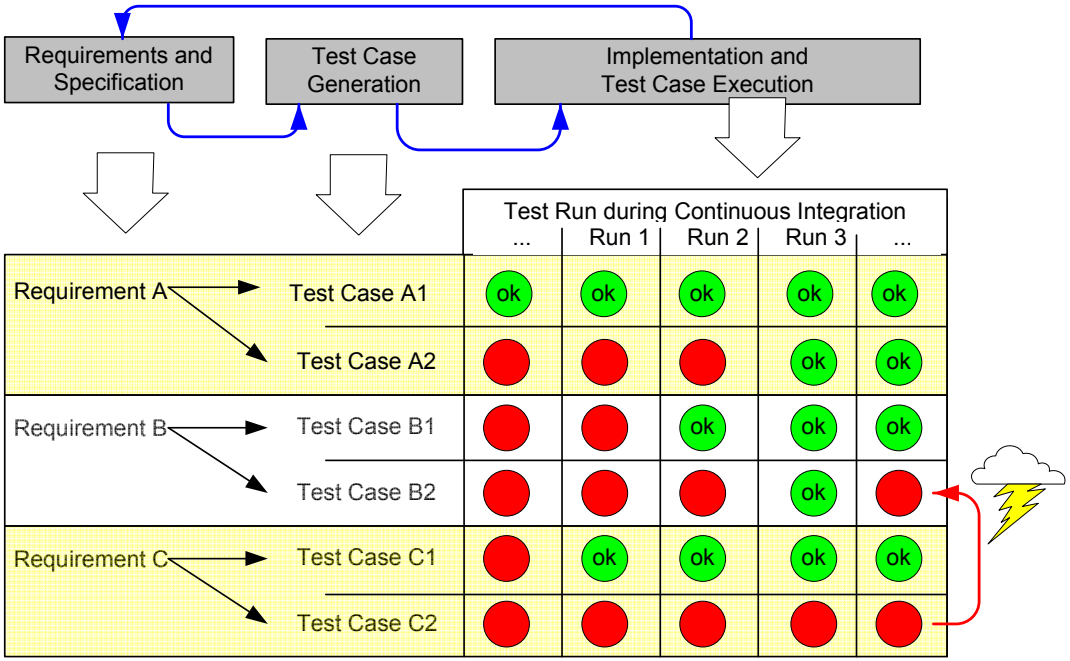
Test-Driven Development Steps:

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.
4. **Refactor** existing code without modifying functional behaviour and test case execution. Continue at step 1.



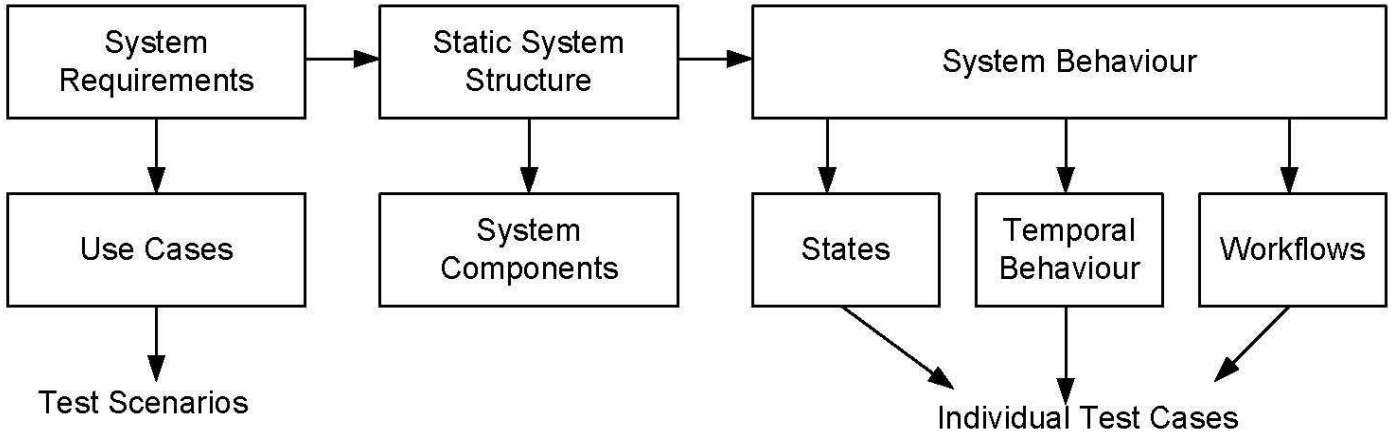
→ **Continuous Integration and Test:**

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



D. Winkler, R. Hametner, S. Biffl: „Automation Component Aspects for Efficient Unit Testing“, Proceedings of 14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Mallorca, Spain, 2009.

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



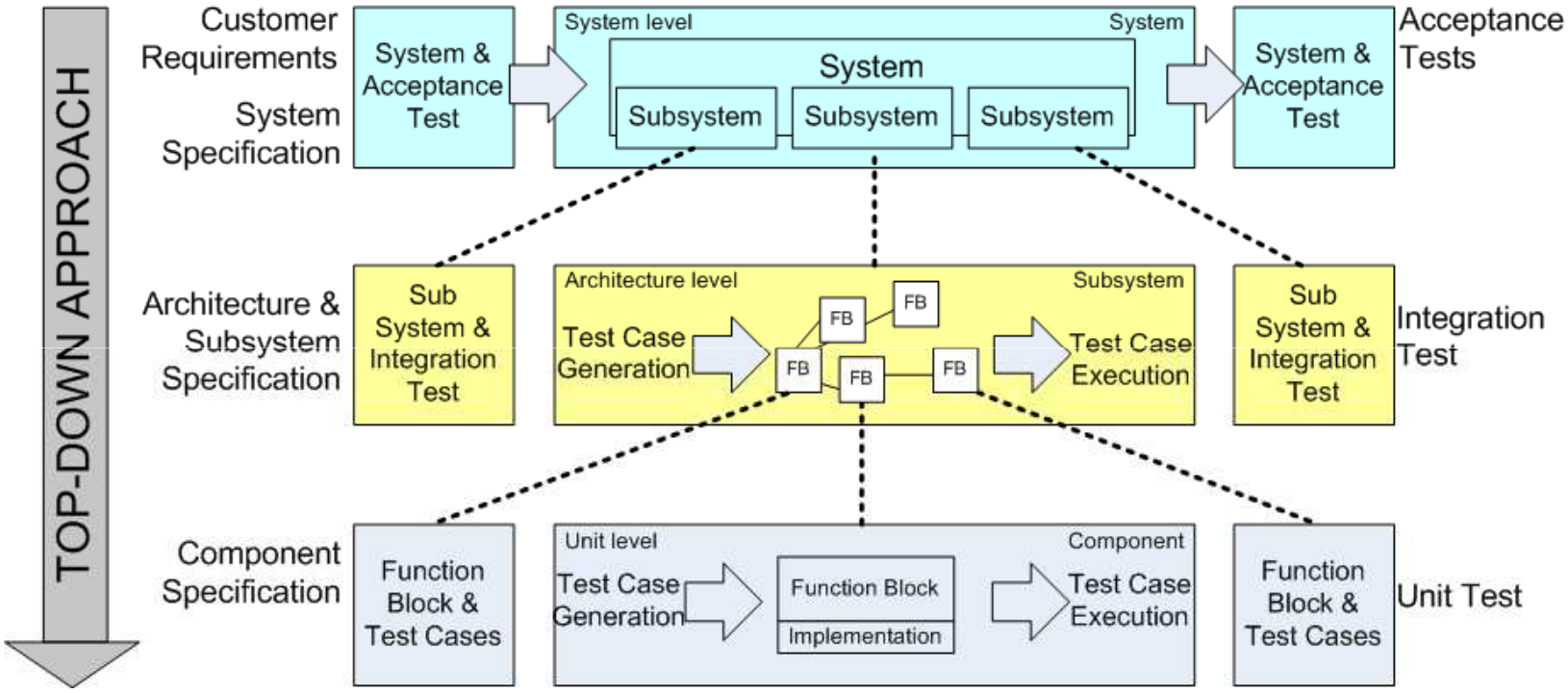
Phase	UML Diagram Type	Test Level	Stakeholder
Requirements Definition	Use Cases Activity Diagram	System / Acceptance Testing	Customer, Systems Integrators
Functional and Technical Systems Design	Deployment Diagram Component Diagram State Chart	Architecture / Integration Testing	Engineering Team
Component Specification	State Chart Sequence Chart Timing Diagram	Component Testing	Individual Engineer
Implementation	State Chart Sequence Chart	Developer Testing	Individual Engineer

R. Hametner, D. Winkler, T. Östreicher, N. Surnic, S. Biffl : „Selecting UML Models for Test-Driven Development along the Automation Systems Engineering Process“, 15th IEEE Int. Conf on Emerging Technologies and Factory Automation (ETFA), WIP, Bilbao, Spain, 2010.

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

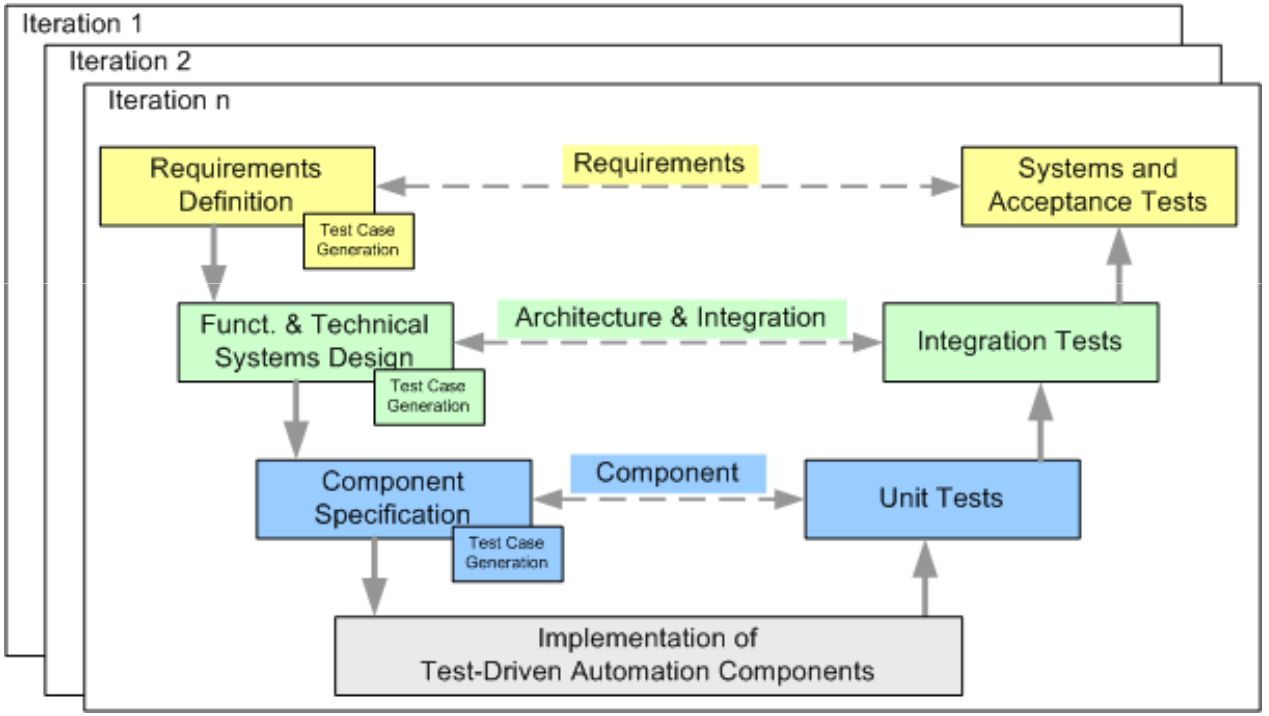
Research Approach

Need

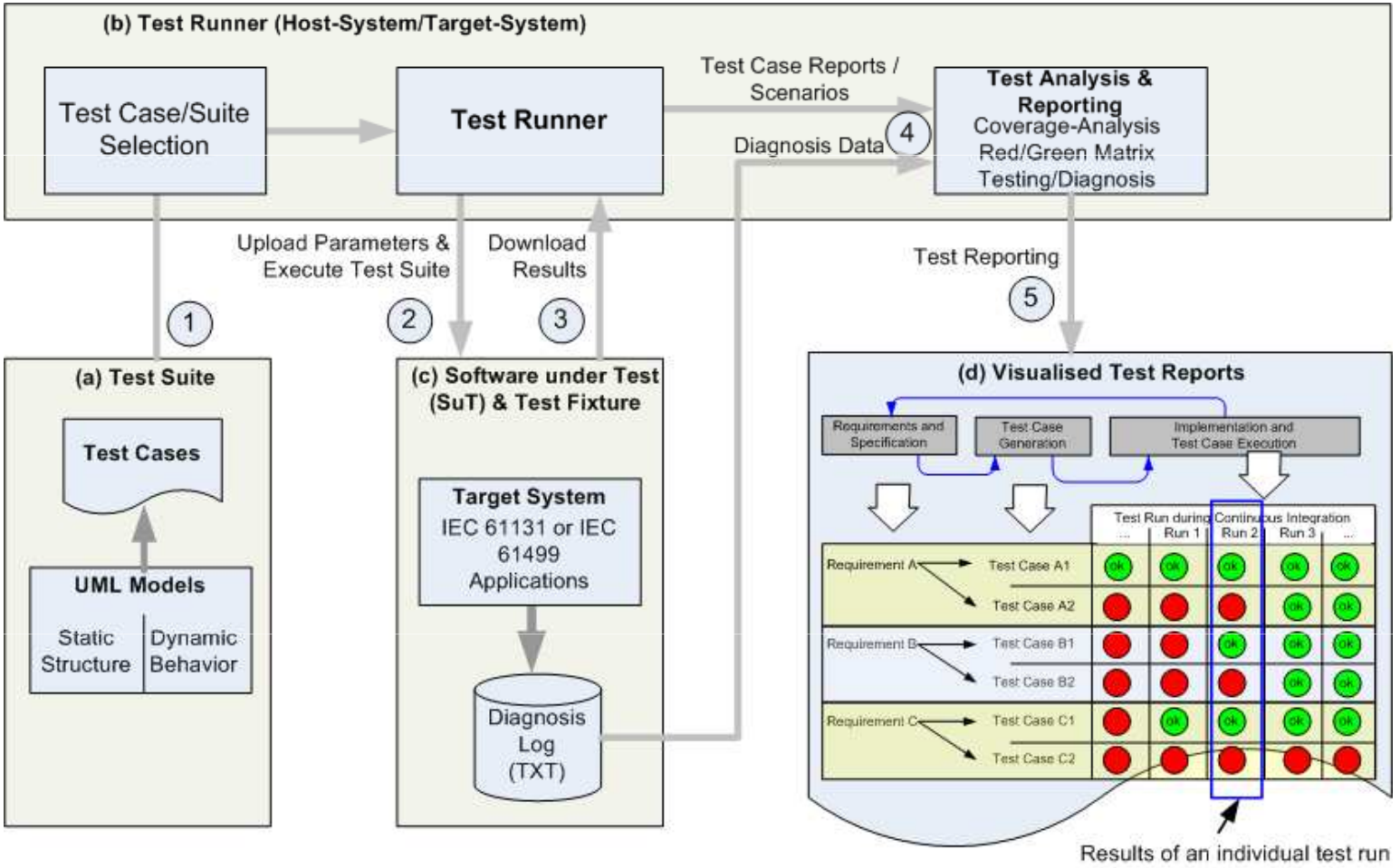
- Automation supported and flexible systems development processes and efficient testing in automation systems projects.

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.



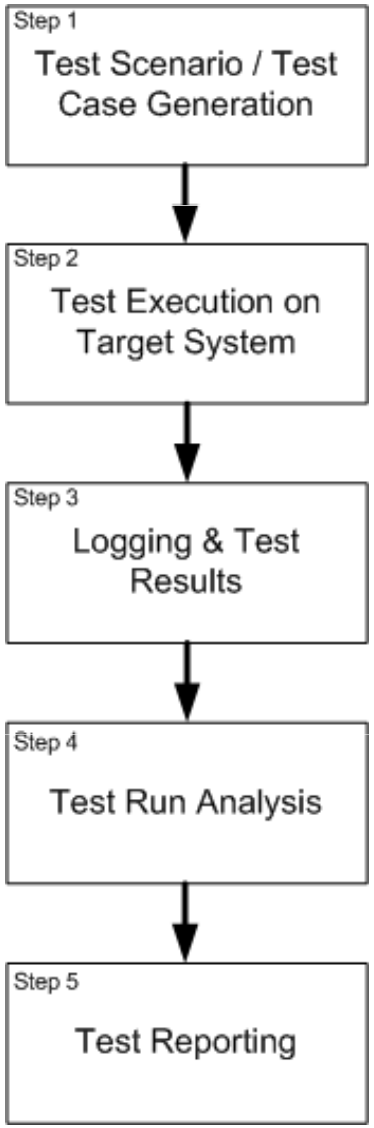
Test Framework for Test-First Development of Automation Systems



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

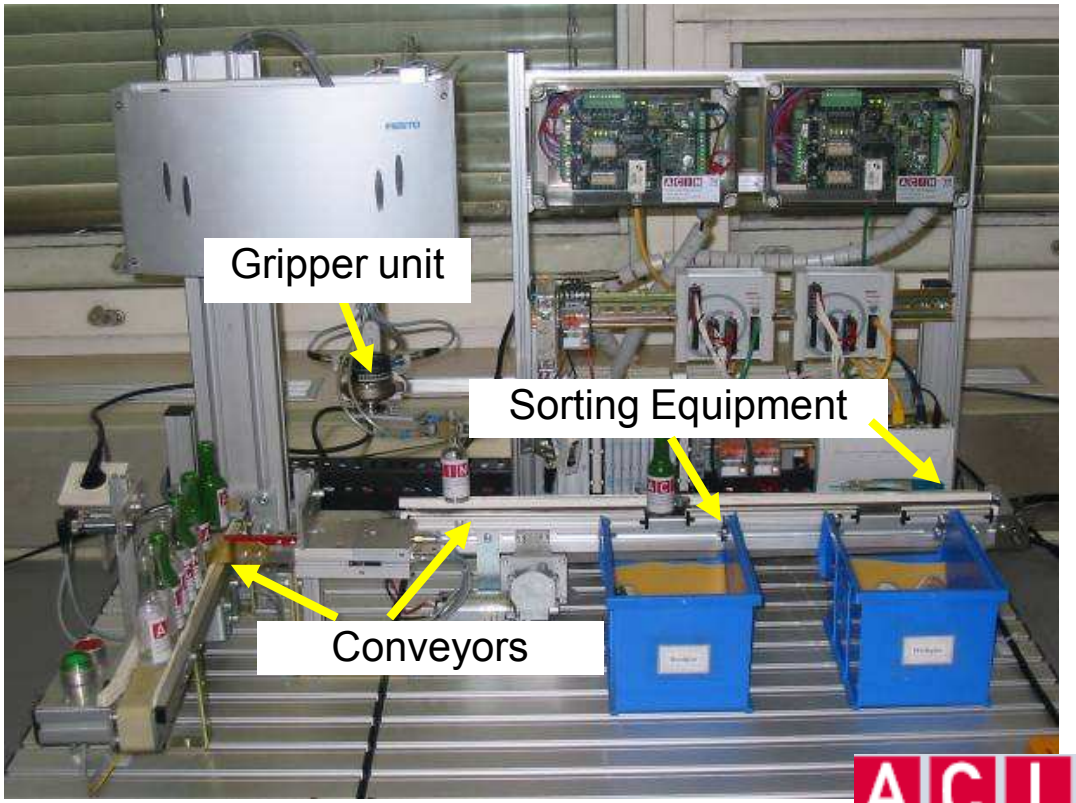
Test Process

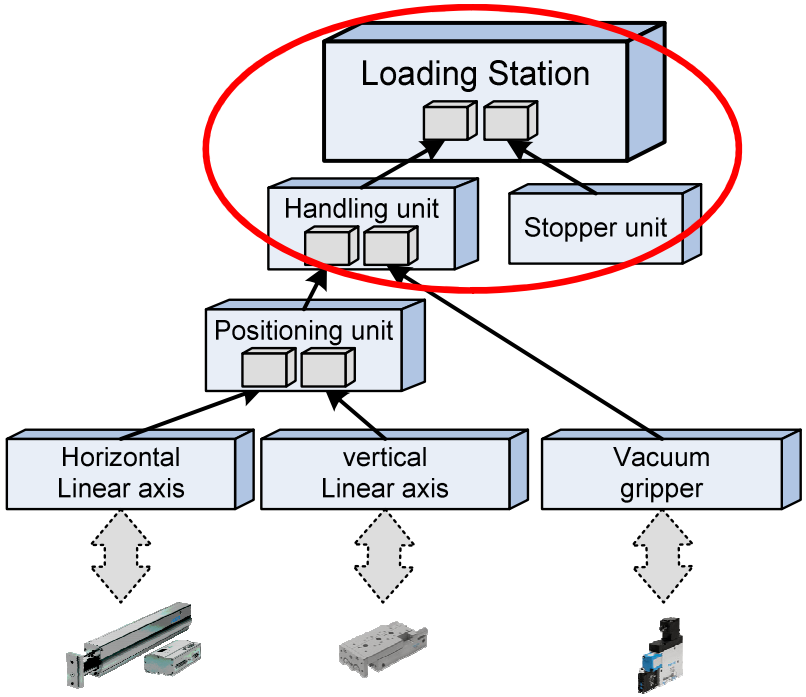
- **Step 1: Automation supported test case generation**
  a) Capturing basic systems requirements.
  b) Test Scenarios based on Use Cases.
  c) Automation supported test case generation.
  d) Test-Framework with keyword driven test.
- **Step 2: Test case execution on target system**
  a) Upload code 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.





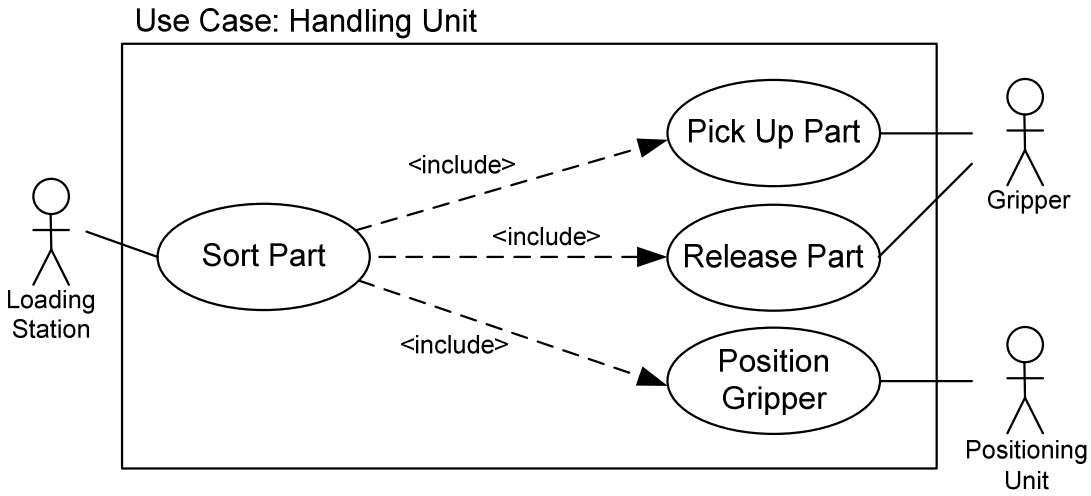
UML Component diagram of the Handling unit

Sorting Application Prototype:
Test Scenarios based on Use Cases

- 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:

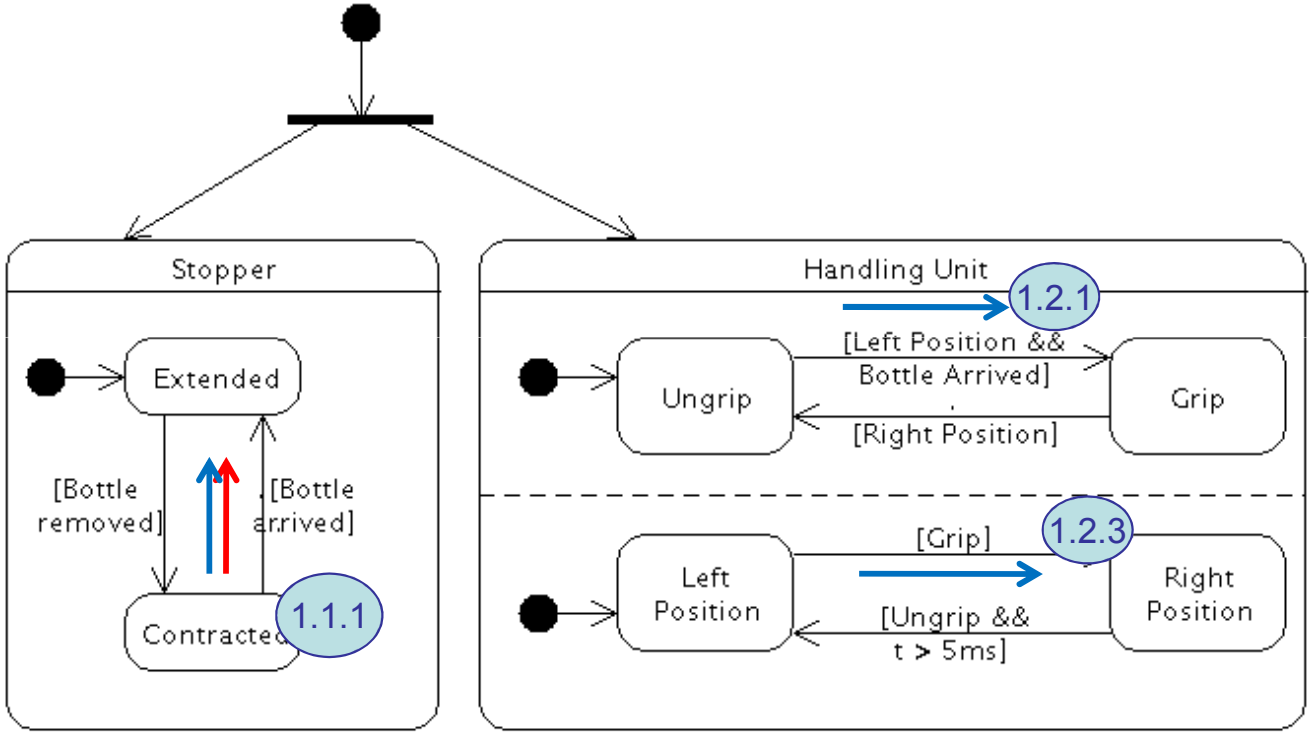
- Common “language” between different disciplines.
- Enhanced understanding of the customer requirements.
- Test scenarios as vehicle for communication between stakeholders



No	Description	Scope	Pre-condition	Action	Expected Result / Post-Condition
1	Sorting a Bottle	System: Bottle sorting application	Handling Unit in idle position No bottle present	Command to sort bottle	Handling unit in idle position and part sorted
1.1	Recognizing Bottle at Conveyor 1	Subsystem: Stopper	Conveyor running Bottle available	Stopping bottle	Bottle stopped by stopper unit
1.2	Moving Bottle from Conveyor 1 to 2	Subsystem: Handling Unit	Bottle available Gripper in idle position	Gripping, moving, and releasing bottle	Bottle moved to conveyor 2 Gripper returned to idle position

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.



Executed Test Case

Test Scenario with three Test Cases

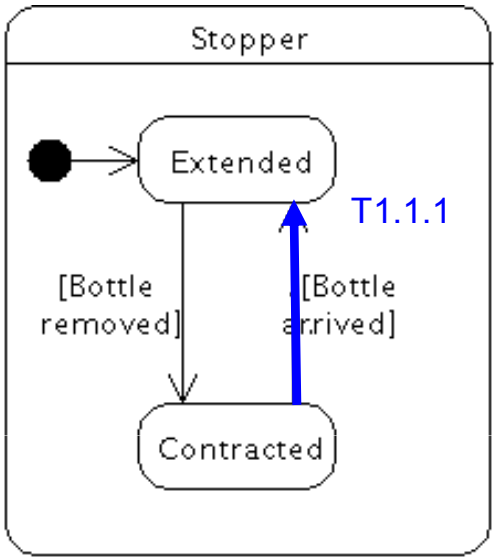
Sorting Application Prototype:
Derived Test Cases from State Chart

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

No	Description	Pre-condition	Action	Expected Result / Post-Condition
1.1.1	Stopper Unit: Stop bottle	State=Stopper.Contracted No bottle present	Bottle arrived	Stopper extended && bottle stopped
1.1.2	Stopper Unit: Bottle removed	State=Stopper. Extended Bottle present	Bottle removed	Stopper contracted
1.2.1	Handling Unit: Grip bottle	State=(Stopper.Extended && HandlingUnit.Ungrip && HandlingUnit.Left)	Bottle arrived	State=(Stopper.Extended && HandlingUnit.Grip)
1.2.2	Handling Unit: Release bottle	State=HandlingUnit.Right && HandlingUnit.Grip	Ungrip	State=HandlingUnit.Right && HandlingUnit.Ungrip
1.2.3	Handling Unit: Move and release bottle	State=(Stopper.Extended && HandlingUnit.Left)	Move to Right	State=(Stopper.Extended && HandlingUnit.Grip && HandlingUnit.Right)
1.2.4	Handling Unit: Return to idle state	State=HandlingUnit.Right	Wait 8ms && Ungrip	State=HandlingUnit.Left

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.



	A	B	C
1	Script	TEST_IDLE	
2	P1		
3	P2		
4	P3		
5	P4		
6	P5		
7	Res	WAHR	
8			
9			
10	Run Test		
11			
12	Ergebnis	Zyklus/Zeit	Aktion
13	WAHR	0	START
14	WAHR	1	FORCE
15	WAHR	1	FORCE
16	WAHR	1	FORCE
17	WAHR	1	FORCE
18	WAHR	1	WAIT
19	WAHR	1	ASSERT
20	WAHR	1	ASSERT
21	WAHR	1	FORCE
22	WAHR	1	WAIT
23	WAHR	1	ASSERT
24	WAHR	5	ASSERT
25	WAHR	1	ASSERT
26	WAHR	1	ASSERT
27	WAHR	5	STOP
28	Anweisungen vor dieser Zeile einfügen		

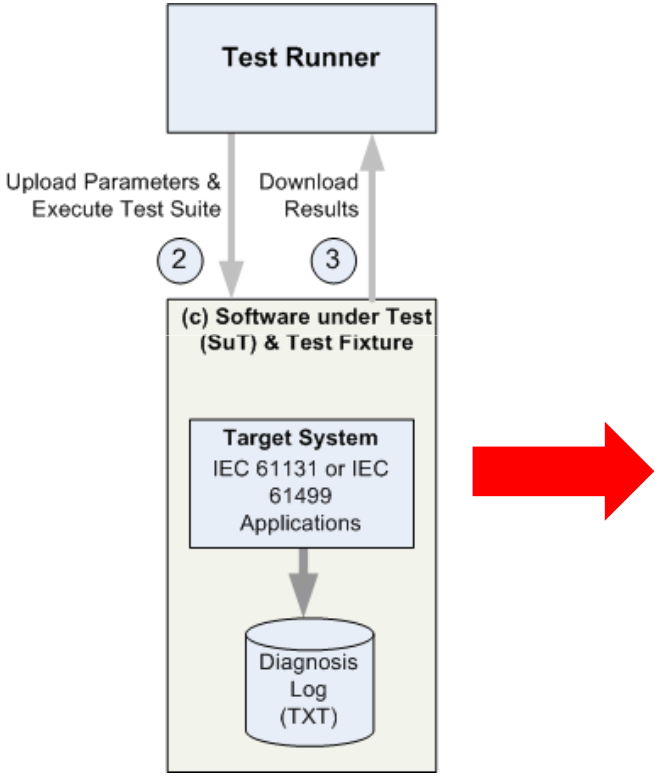
	A	C	D	F	I
1	Resource	Offline-Conf/Offline-Tests		Mode	
2	Type of Test	OFFLINE			
3					
4	Run Test				
5					
6					
7	System / Component	Test Id	Scenario	Pre-Condition	Status
8	Sorting Application	T1	Sorting a bottle	System is running	WAHR
9	Stopper Unit	T1.1	Recognizing Bottle at Conveyor 1	Conveyor running, bottles available	WAHR
10	Stopper Unit	T1.1.1	Stop bottle	Stopper.contracted	WAHR
11	Stopper Unit	T1.1.2	Bottle removed	Stopper.Extended	WAHR
12	Handling Unit	T1.2	Moving bottle from Conveyor 1 to	bottle available, gripper idle	WAHR
13	Handling Unit	T1.2.1	Grip bottle	State=(Stopper.Extended && HandlingUnit.Ungrip && HandlingUnit.Left)	WAHR

	D	E	F	G	H	I	J	K
13								
14	ACTIVATE		FALSE					
15	A		FALSE					
16	B		FALSE					
17	TimeDiscrepancy		10					
18		1000						
19	READY		FALSE					
20	DiagCode		0x8000					
21	ACTIVATE		TRUE					
22		1000						
23	READY		TRUE					
24	SF_EquivalentOut		FALSE					
25	Error		FALSE					
26	DiagCode		0x8001					
27								
28			Anweisungen vor dieser Zeile einfügen					

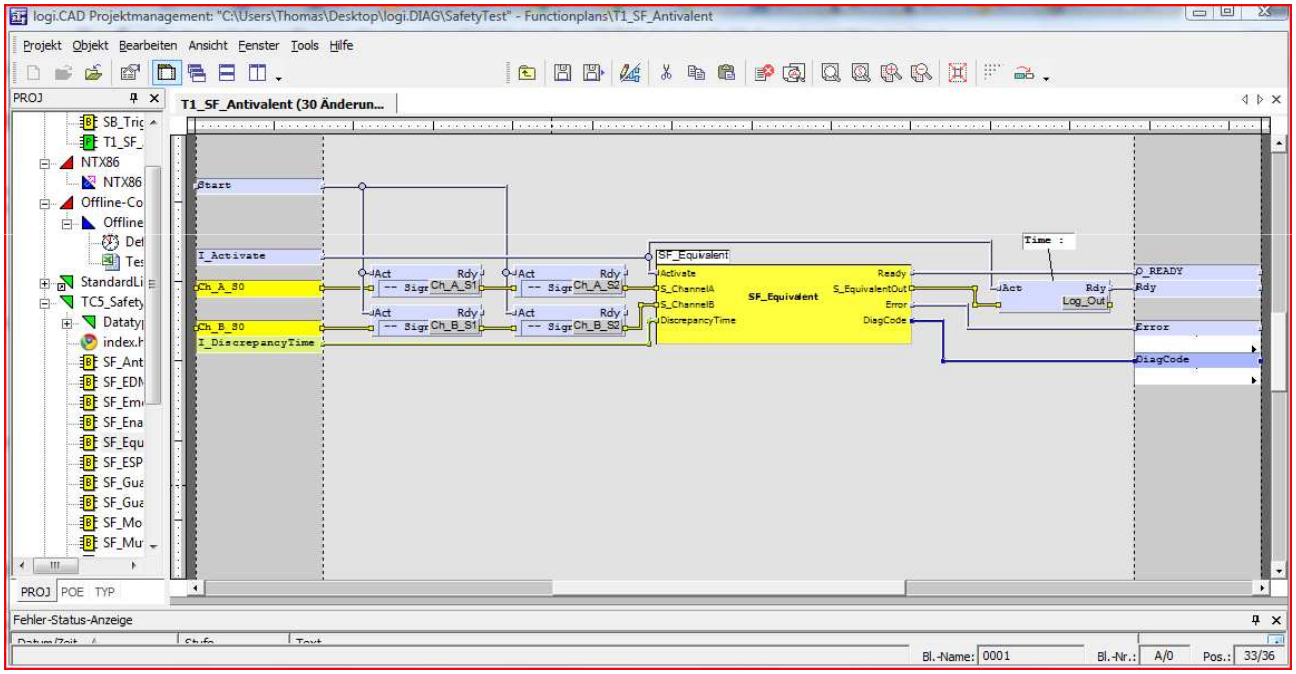
Sorting Application Prototype:
Capture test and diagnosis results

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



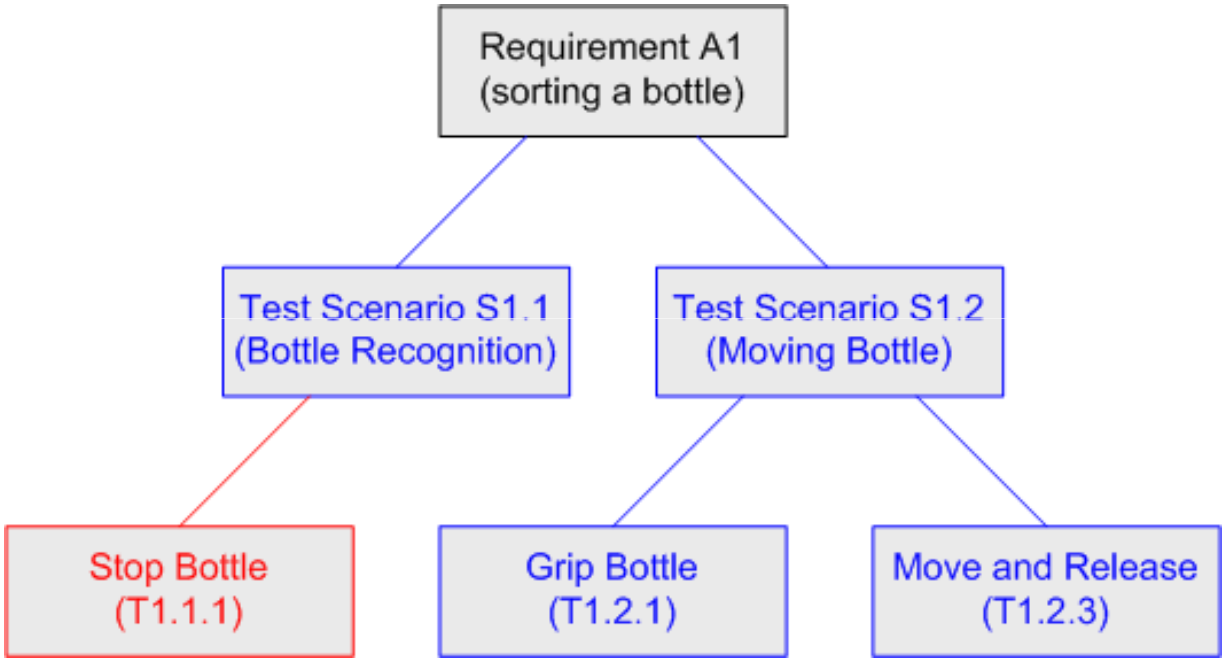
Test Runner



Sample Test Case Execution on Target System with logi.CAD/RTS

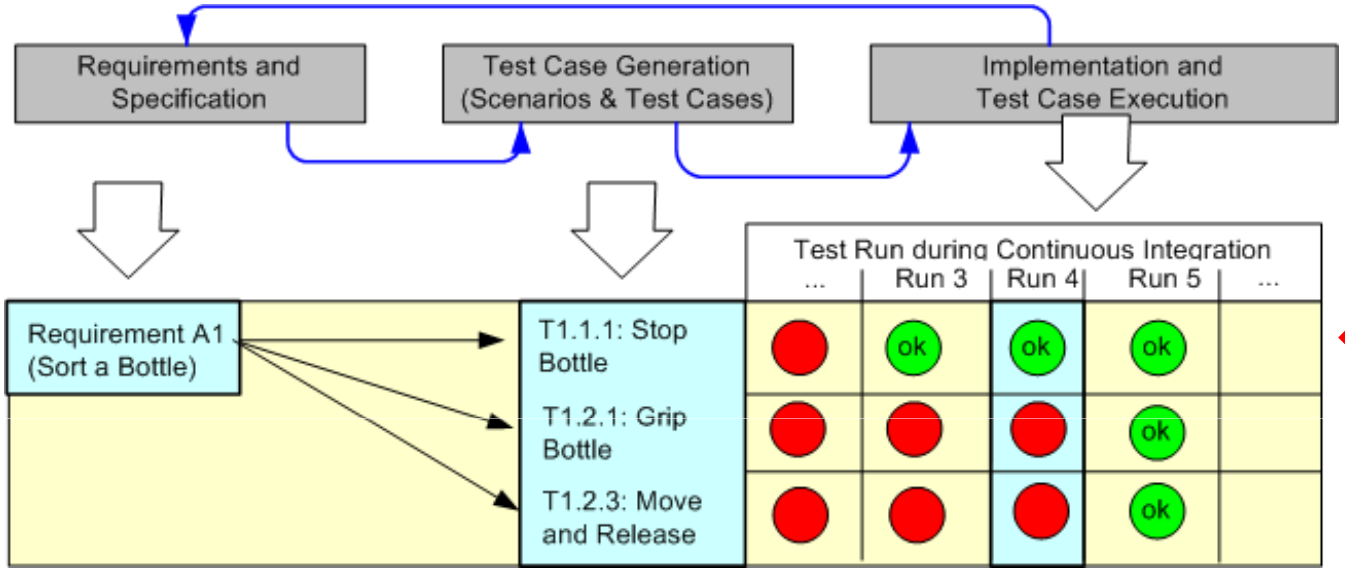
Sorting Application Prototype:
Analyzing Test Results

- 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.



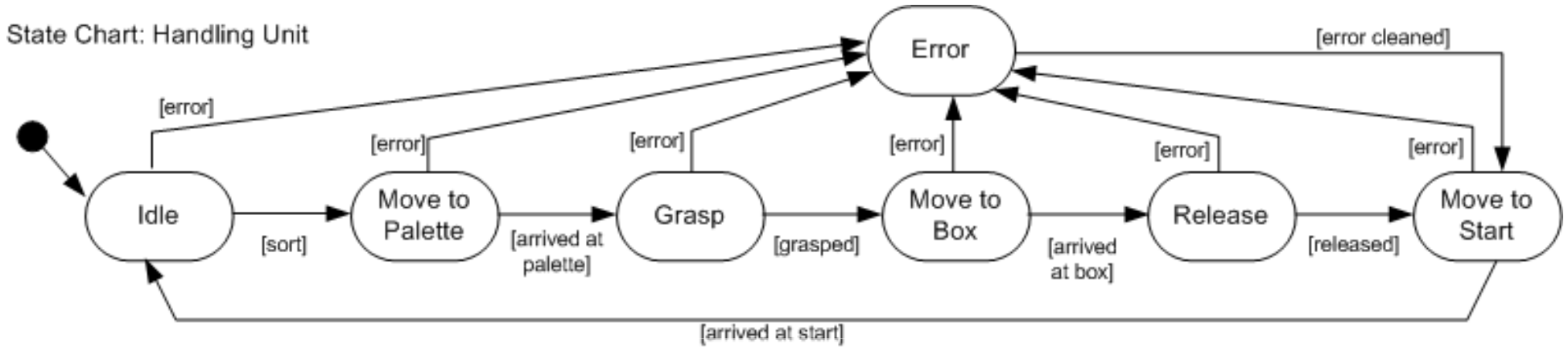
Lessons Learned & Future Work

- 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

Sorting Application Prototype:
Integration and Unit Tests

- 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.



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