The Future is Software

The Vector Tool Strategy for Software Testing
<table>
<thead>
<tr>
<th></th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Motivation</td>
</tr>
<tr>
<td>2.</td>
<td>Test Concepts</td>
</tr>
<tr>
<td>3.</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Motivation

The Future is Software!

Innovation & high added value

Combustion Engine
Mechanics
Electronics
Software

1886 1920 1980 2025

Time

Smart Home  Automotive  Embedded System
Industry 4.0  Avionics
Medical  Railway
Cyber Physical Systems

**Motivation**

**Cyber Physical System**
Electrical or mechanical system connected to the real world controlled by software (algorithms)

**Physical Environment**
Mechanics/electric connected to the control system via sensors and actuators

**Control System**
Software controlling and monitoring the cyber physical system

**System under Test**
Software system to be developed and tested, built out of subsystems and software components

**Software System Environment**
Software not in the focus of the system to be implemented and tested
Challenges and Goals

- Test the software early in the development process
- Test the software independent of real hardware and HIL test rigs
- Support testing of highly distributed systems
- Seamless tool chain and test design methods
- Test components independent of availability of other components
- Highly parallel testing
- Seamless tool chain and test design methods
- Test the software early in the development process
- Test the software independent of real hardware and HIL test rigs
- Support testing of highly distributed systems
- Highly parallel testing
1. Motivation
2. Test Concepts
3. Conclusion
Deliver Incrementally

“Walking on water and developing software from a specification are easy if both are frozen.”
Edward V. Berard
Software Quality Gates

Solution:
- Software quality gates
- Automated at each stage $T(x)$
  - **Isolate**: Target each developer commit – $T(0)$ – “Time Zero”
  - **Verify**: Functionality, quality metrics, checks for completeness, traceability, …
  - **Merge**: $T(0)$: dev desk $\rightarrow$ $T(1)$: feature branch $\rightarrow$ $T(2)$: dev branch $\rightarrow$ $T(3)$: release branch

Dev branch $\rightarrow$ Release branch $\rightarrow$ $T(x)$ $\rightarrow$ $T(x+1)$
Test Concepts

Hybrid Test Design: Test Scripts & Models

Control the models
- Parameterization
- Variant selection
- Fault injection

Test script to verify the behavior of the SUT
- Sequentially stimulate SUT
- Observe SUT

Test Script

SUT

Models
- Physical Environment & Software Environment

Control System

Models to integrate the SUT into its "natural" environment
- Asynchronous interaction between models and SUT
- Models may often be very simple, only testing relevant part are modelled
Decomposition and Testing

“Divide and Conquer” Approach:

- SUT is build out of subsystems and software components (features) → divide
- When decomposing/partitioning the SUT in a concrete test setup, only a single component or subsystem will be tested → conquer

Integration workflow for SWC A:

- Step 1 – full decomposition
  - Test SWC A in isolation (decomposed SUT)
  - Other parts of SUT (SWC B, C, D, E) are simulated appropriately
  - Simulate system and plant environment appropriately
- Step 2 – subsystem integrated
  - Test subsystem 1 (decomposed SUT)
  - Subsystem 2 is simulated appropriately
  - Simulate system and plant environment appropriately
- Step 3 – SUT fully integrated
  - Test the whole SUT
  - No parts of SUT are simulated
  - Simulate system and plant environment appropriately
Test Concepts

Virtual Execution Environment

**SUT:**
All kind of software
- Windows, Linux
- ROS, POSIX, ...

**Virtual execution environment:**
- Deploy and execute SUT in a virtual environment
- Hosted
  - on PC hardware
  - in a virtual machine
  - in the cloud
Change-Based Testing

Test Cases
(unit, integration and system tests)

SUT Source Code

Change-Based Testing:
- Source code traceability: Which source code is covered by which test case?
- Which test cases are affected by a source code change?
- Only test what has changed
  - reduce test time
  - find errors faster
  - improve testing efficiency
Test Concepts

Integration Workflow

Isolated Component

Subsystem Integrated

Fully Integrated SW

- SWC A
- SWC C
- SWC B

- Subsystem 1
- Subsystem 2

- Software SUT

- System Env. & Plant Models
  - Software Environment Model
  - Physical Environment Model

Test Scripts
- Functional system tests, unit tests,

Subsystems 1, 2

SWCs A, B, C, D, E

...
Vector Testing Solution – Test Tools for all Test Phases

System Validation

Black-Box Testing + Code Coverage on Target
CANoe, vTESTstudio, VectorCAST/QA, VT System

HW/SW Integration Test

Black-Box Testing + Code Coverage on Host
CANoe, vTESTstudio, VectorCAST/QA, vVIRTUALtarget

SW System Test

White-Box Testing + Code Coverage on Host & Target
VectorCAST/C++, VectorCAST/RSP

Link to Requirements

SW Integration Test

SW Unit Test

SW Implementation

Test Concepts

14
Conclusion

How to Meet Challenges and Goals

**Challenges and Goals:**

- Seamless tool chain and test design methods
- Highly parallel testing
- Test components independent of availability of other components
- Support testing of highly distributed systems
- Test the software early in the development process
- Test the software independent from real hardware and HIL racks

**Solution:**

- Software quality gates
- Hybrid test design: test scripts & models
- Decomposition: Divide and Conquer
- Virtual execution environment
- Change-Based Testing
- One consistent tool chain for all test levels
For more information about Vector and our products please visit

www.vector.com

Author:
Katranski, Ute
Vector Germany