Towards Continuous Software Development for AUTOSAR Systems

EMCC 2020 Virtual Edition, 30th of July 2020
Agenda

- **Introduction**
- Applying Cx to AUTOSAR Development
- AUTOSAR Continuous Integration
- AUTOSAR Continuous Delivery
- Summary & Outlook
Tesla Financial Report Q4/2019:

All Tesla vehicles with our FSD computer have been updated with new software that can better detect new details in their environments [...]. We are currently validating this functionality before releasing to customers, and we look forward to its gradual deployment. We also introduced in-app purchases, where our customers can buy various software updates, such as basic Autopilot, Full Self Driving […]. Software will continue to play a growing role in our business model.

[Source: Tesla:"Q4 and FY2019 Update"]
Maturity Levels of Continuous Software Development

Introduction

Plan → Code

Agile

SCRUM, Feature Driven Development, etc.

Feature backlog with ranked tasks

Sprint based feature planning
Maturity Levels of Continuous Software Development

Introduction

... as before, + Server based system with:

- Automatic software configuration
- Software generation (ASW, RTE, BSW)
- Software compilation
- Software linking
- Unit-testing (Integration Tests)
- Short feedback channel for developers
Introduction

Maturity Levels of Continuous Software Development

... as before, + Server based deployment system with:

- Automatic functional tests
- Automatic non-functional tests
- Automatic regression tests
- Automatic acceptance tests
- Product packing (executable, docu, release-notes, ...)
- Rollout to staging area
Introduction

Maturity Levels of Continuous Software Development

... as before, +

automatic software deployment to fleet
Introduction

Maturity Levels of Continuous Software Development

Feedback

Plan → Code → Build → Test

- Agile
- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Continuous Development & Operation (DevOps)

... as before, +

automatic collection of feedback from operation

Including feedback in product planning
### What is Continuous Software Development in Automotive Industry about?

<table>
<thead>
<tr>
<th>Traditional Software Development</th>
<th>Continuous Software Development (Cx)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Update</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Planning</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Feature Structure</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Stop of Planning</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Release</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Good approach for stable times & highly reliable products**

**Good approach for dynamic times with many changes (e.g. technological) & flexible customers**
Agenda

- Introduction
- **Applying Cx to AUTOSAR Development**
  - AUTOSAR Continuous Integration
  - AUTOSAR Continuous Delivery
- Summary & Outlook
Where to start with Cx

... where it brings the most benefit

- When activities are done with a high frequency, e.g. per commit
- When activities are time-critical, e.g. before release/testing
- When activities are repetitive and error prone
- (Duration/Effort is relative – depends on time criticality)
Applying Cx to AUTOSAR Development

Where to focus at AUTOSAR Projects to advance in Cx

<table>
<thead>
<tr>
<th>State of the art</th>
<th>Our focus for this talk</th>
<th>For future</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plan</td>
<td>Code</td>
<td>Build</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>Validate &amp; Pack</td>
</tr>
<tr>
<td></td>
<td>Release</td>
<td>Operate</td>
</tr>
</tbody>
</table>

I – AUTOSAR Continuous Integration

- Cross-Team/Organizational Development
  - Integration of 3rd Party Software
- ASW to BSW integration
  - Manual job by integrator (often bottleneck for executable software)
- Providing fast testing possibilities

II – AUTOSAR Continuous Delivery

- Checking timing requirements
  - e.g. meeting reaction constraints
- Validating capacity boundaries
  - Staying inside predefined boundaries for software parts / ECU resources (CPU, memory, stack)
- (functional testing is not considered here)
Agenda

- Introduction
- Applying Cx to AUTOSAR Development
- **AUTOSAR Continuous Integration**
- AUTOSAR Continuous Delivery
- Summary & Outlook
For Developers
- Getting testable software in hours instead of weeks
- Reducing number of manual & monotonic tasks

For Integrators
- Reducing repetitive tasks and focus on more complex system integrations (conflicts)

For Project Manager
- Having early & continuous transparency on progress of implemented functionality

Objectives for Continuous Integration

AUTOSAR Continuous Integration

Continuous Integration

Feedback

Plan  Code  Build  Test

Efficiency
Speed
Change of Mindset
Major Concepts for AUTOSAR Continuous Integration

- Eliminating exclusive operations on central artefacts
  - Preventing blocking of other engineers

- Applying Decision Frontloading for integration
  - For automatizing integration activities

- Use virtual testing
  - Executing Smoke Test for minimal executability check
Current Approach

- **System Architect**
  - Updates Communication-Matrix or Diagnostic Description

- **SW Developer**
  - Modifies SWC description according to changed software

- **Integrator**
  - Applies all changes by integrating application software to BSW
    - Runnable to Task mapping
    - Runnable Positioning inside Task
    - Port Mapping

Integrator is the bottleneck for SW Developers
AUTOSAR Continuous Integration

Cross-Team/Organizational Development

New Approach

- **ECU-C (Base)**
- **ECU-C (Root)**
- **System Description**
- **Diagnostic Description**
- **Communication Description**
- **SWC Description**
- **SW Developer**
  - Modify & implement
- **Modify & implement**

SWC Description (cutout) modified
Integration Instr.

Auto-Integration

Executable
with local changes

Clone

Build

ECU SW Architect/Integrator

Organize

BSW/Platform Expert

SWC Description (cutout) modified
Integration Instr.

Auto-Integration

Executable
with all changes

Build

ECU-C Root’
AUTOSAR Continuous Integration

Workflow

App Package
- SWC (atomic or composition)
- Integration Instructions
- Implementation (Source-Code)

Root Configuration
- DPA project configured for the used ECU, without apps and RTE

Option CI

Integration Result
- RTE configuration based on integration Instructions
- App-dependent BSW configuration
Integration Instructions for CI

Major activities in case of changed Application Software

1. Runnable to Task Mapping & Positioning

2. Port Mapping

3. Data Mapping
1. Via Direct Integration Instructions
   ▶ Explicit Runnable to Task Mapping

2. Via Abstract Integration Instructions
   ▶ Describing only in abstract way
     ▶ Activation Period
     ▶ Safety Level
   ▶ Mapping via matching properties

3. Heuristic Based
   ▶ Using AUTOSAR model information
     ▶ Runnable Trigger
     ▶ Safety Level
     ▶ Data Dependencies
     ▶ Resource Needs
   ▶ Matching to Task definition via solving algorithm
ASW Integration Instruction for CI – Runnable Positioning

1. Via Direct Integration Instructions
   ▶ Defining Execution Sequence of all Runnables

2. Via formal Constraints
   ▶ Using Execution Order Constraints or Data Age Constraints for describing time critical dependencies between Runnables

3. Via Heuristics (Auto-Map)
   ▶ Analyzing dependencies between Runnables and minimizing backward references
ASW Integration Instruction for CI – Port Mapping

1. Direct mapping via Integration Instructions
   - Mapping SWC port to
     > Ports of other SWC
     > Service ports (e.g. Trigger Ports, ...)
   - Non-existing SWCs and Ports in integrated App Packages can be stubbed

2. Via Heuristics (Auto-Map)
   - Matching of Port Names
Using DaVinci Configurator Pro.CI in Agile Development Process

- ECU Extract Updates get integrated in Root Configuration at dedicated dates (e.g. till sprint start)

- App Developers work during sprint on prepared Root Config (Baseline)
  - In case of urgent ECU Extract Updates, a manual Baseline Update is required

- Commit of App Package to Build Server produces Executable, based on specified Baseline
Result  3 Types of Builds (offered by Option .CI)

1. Virtualization on VFB Level (vVIRTUALtarget Pro)
   - For testing ASW functions standalone (e.g. for Unit-Tests)

2. Virtualization on BSW Level (vVIRTUALtarget Basic)
   - For testing ASW functions in combination with BSW (e.g. for Integration-Test with BSW)

3. Hardware
   - For functional HIL tests
   - For trace measurements (see later)
Agenda

- Introduction
- Applying Cx to AUTOSAR Development
- AUTOSAR Continuous Integration
- **AUTOSAR Continuous Delivery**
- Summary & Outlook
AUTOSAR Continuous Delivery

Objectives for Continuous Delivery

For Developers & Testers
- Validate all functional and most non-functional properties automatically (feedback on malfunction)
- Increasing confidence in committed software

For Project Architects
- Be sure all system and function specific timing requirements have been met
- Getting early notice of potential future resource exceeds

For Project Manager
- Be sure that all resource restrictions have been met
- Making software directly deliverable
Major Concepts for AUTOSAR Continuous Delivery (CD)

- Formalize Performance & Timing Requirements
  - For executing non-functional tests automatically

- Perform Model-based Timing & Performance Analysis
  - For having fast feedback and being not limited to physical resources during tests
AUTOSAR Continuous Delivery

CPU Resource Requirements

Application Software Requirements
- e.g. utilization boundaries for Compositions, Software Components or Runnables

Hardware Requirements
- e.g. maximum core utilization
Timing Requirements

Application Software Requirements
- e.g. maximum execution time for Runnables
- e.g. data ages for communication between Runnables

Operating System Service Requirements
- e.g. Task response times
- e.g. Start-to-Start durations

System Requirements
- e.g. Event-Chains on critical dataflow / End-to-End times
Objectives for Testing

In contrast to normal (non CD) testing ...

- Testing should cover many different Scenarios
  - e.g. Burst-modes with many service-requests
  - e.g. increased load scenario, when function amount increases
  - e.g. different execution modes
- Fast test execution is required
  - Tests should run in hours (max. a day) instead of weeks

Simulation allows fast exploration
Types of Timing & Performance Validation

1. Model-based Simulation
   - Not limited to hardware, therefore theoretical unlimited parallelization
   - For Stress-tests, system stimulation scenarios can be arbitrary extended in order to cover many situations
     - Requires execution times for Runnables

2. Hardware Trace Measurements
   - Allows detailed analysis of error scenarios on unlimited level of granularity
   - Execution times can be measured
     - Limited through physical hardware
AUTOSAR Continuous Delivery

Procedure

Build → ECU → Hardware Measurement → Model-based Simulation → Timing Report

Runnable Execution Times

Parametrized random parameter generator

System Description → (alternative) Supplement Runnable Execution Times to SysDescr

ECU-C

Test Report
Hardware Measurement Overview

1. Configure MICROsAR Tracing Hooks (OS and RTE)
2. Configure iSYSTEM
3. Trace the ECU and export a BTF
4. Evaluate the Traces against Requirements
Example of Timing Report

- Showing measurement, tool and input file information
- Validating system requirements (meet/failed)
- Showing further timing metrics for later evaluation
- Inputs such as Task Stack Consumption can also be stored
JUnit Reports and Continuous Timing Evaluation

- Junit Reports for automated testing
- Publish direct on the CI Server
- Combination with other testing results in one overview

- Showing CPU Load average, as well as minimum and maximum value (in a x ms timeframe)
- Extrapolation shows CPU Limit will potentially be reached within two releases

Performance improvements should be applied now, in order to prevent a blocking of delivery
Agenda

- Introduction
- Applying Cx to AUTOSAR Development
- AUTOSAR Continuous Integration
- AUTOSAR Continuous Delivery
- **Summary & Outlook**
Summary & Outlook

Conclusion

▶ CI brings already a lot of benefits for the agile software development, especially for developers and integrators
  ▶ Decision Frontloading enables fast integration
  ▶ Independent local integration solves blocking
▶ CD makes sense, when software is often released (e.g. sprint based, ~quarter based)
  ▶ Automatic non-functional testing, e.g. timing or performance is essential

What’s Next?

▶ Standardization of Timing Hardware Measurement (AUTOSAR ARTI)
▶ Over-the-Air Updates (OTA)
For more information about Vector and our products please visit

www.vector.com

Author: Deubzer, Michael
Vector Germany