AUTOSAR tool implementing AUTOSAR methodology
- And Some Customizations Introduced

Jeongho Son, HYUNDAI AUTRON
7th AUTOSAR Open Conference
Detroit, October 22nd 2014
Contents

1. Review
2. Tool implementing AUTOSAR Methodology
   - Screenshots Continued
3. Customizing Methodology
4. Integrated Development Environment
5. Future Work
6. Summary
1. Review

AUTOSAR Methodology
AUTOSAR Software Architecture

Tool support required Everywhere!

- Software Component
- Network Import
- ECU Mapping/Extract

- RTE Configuration
- BSW Configuration
- Generating ECUC based on Upper Templates
- Code Generation
2. Tool Implementing AUTOSAR Methodology

Software Architecture Design: Data Type

• Application Data Types: Primitive / Array / Record
• Implementation Data Type
• Sw Base Type
• Compu Method
Software Architecture Design: Port Interface

- Client Server Interface
- Sender Receiver Interface
- Parameter Interface
Software Architecture Design: Composition
Software Architecture Design: Composition

Features:
• **Direct Port Editing**: Addition, Removal, Rename of Ports
• **Drag-And-Drop Creation**:
  - Port Prototypes from Port Interfaces
  - Component Prototypes from Component Type
• **Highlighting Compatible Ports**
Component Type

- **Runnable**: Easiest Addition/Removal of Data Access Points, RTE Events, Wait Points
- **Exclusive Area**
- **Variable / Parameter**
- **Port**
Advanced Generic Editor: ‘All Contents’ Page

- Editing UI for any AUTOSAR Metamodel
- Consists of Tree/Table Viewer, Field Editors
Import Network Design: CAN DBC ...

* Network Format Supported

<table>
<thead>
<tr>
<th>Type</th>
<th>File Format</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN</td>
<td>DBC</td>
<td>-</td>
</tr>
<tr>
<td>LIN</td>
<td>LDF</td>
<td>2.0 or above</td>
</tr>
<tr>
<td>FlexRay</td>
<td>FIBEX</td>
<td>2.X</td>
</tr>
<tr>
<td>Ethernet</td>
<td>FIBEX</td>
<td>4.X</td>
</tr>
<tr>
<td>Diagnostics</td>
<td>ODX</td>
<td>2.X</td>
</tr>
</tbody>
</table>
ECU Mapping / Extract

- Swc Mapping
- Data Mapping
- ECU Extract
ECU Configuration: Generic ECUC Editor, OS Case

Consists of: Multiple pages, Tree/Table Viewer, Field Editors
One Container Focused: Various Field Editors
Code Generation

Incorporates any command-line executable
3. Customizing Methodology

Software Architecture: Supports Network-Oriented Design Approach

① DBC Import:
In addition to Fibex elements, creating
• Data Type
• Compu Method
• Data Constr
• Port Interface, Data Element

② Software Components:
• Adding Ports
• Adding Data Access Points

③ Data(Signal) Mapping
Framework of ECU Configuration Generation

Key Point:
• BSW module developers themselves write converting codes for their modules.

ECUC Gen Frontend (Wizard, File IO)
Automation Framework
General Automation
Module Automation
ECUC Diff/Merge Framework
Automation Util
AUTOSAR File Util
ARTOP (AUTOSAR SDK)
Eclipse

Invoke
AutoConfig EcuAction

EcucDiff Util
Automation Util
Automation FileUtil
ECU Configuration Generation: Eclipse Extension Point
ECU Configuration Generation Wizard

① Select System Descriptions

② Select ECUC Modules to Fill

③ Check Options
Concept of Platform Delivery Process: CAN Example

**System Desc. from CAN DBC file**

**CanStack Configuration from Platform SRS**

**Input**

**ECUC Gen Wizard**

**Other System Desc ?**

**Platform SRS**: Platform Delivery Order from ECU Software Developers describing Platform Requirements
Stack-wise ECUC Editing: Virtual ECU Module

Example: Memory Stack Editor

Ecuc Module Def: AUTRON_EcucDefs_NvRamStack.arxml

UI Rendering

NvM module

Fee module

Ea module

Set Real Module Values

Callback invoked upon User Editing Input

Editing Handler: ‘NvRam ExtendedEcuc EditingHandler’
Memory Stack Editor: Eclipse Extension Point
Memory Stack Editor: Nv Block Page
4. Integrated Development Environment

Integrated Build System

Features
- SCons based
- Build scripts written in Python
### Current Status of Tool Integration: Eclipse

<table>
<thead>
<tr>
<th>Tool Plug-in</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOSAR Tool</td>
<td>AUTOSAR Authoring/Converting/Generating</td>
</tr>
<tr>
<td>Subversion Client</td>
<td>Configuration Management</td>
</tr>
<tr>
<td>C/C++ Development Tooling (CDT)</td>
<td>C Coding, Build</td>
</tr>
<tr>
<td>WTP Source Editing</td>
<td>XML Editing</td>
</tr>
<tr>
<td>Python IDE (PyDev)</td>
<td>C Build Script Settings</td>
</tr>
<tr>
<td>Plug-in Development Environment</td>
<td>Extending AUTOSAR Functionalities:</td>
</tr>
<tr>
<td></td>
<td>• ECU Configuration Generation</td>
</tr>
<tr>
<td></td>
<td>• ECUC Stack-wise Editor</td>
</tr>
</tbody>
</table>

➤ **Software developers do NOT have to leave ECLIPSE most of the time while doing their software jobs**
5. Future Work

1. Improve Interaction/Interoperability with MBD Tool
   • Import/Export from/to SIMULINK
   • Diff/Merge during Import

2. Extend Coverage of Automated ECU Configuration
   (ECU Configuration Generation)
   • Diagnostics based on ODX file
   • Services such as NvM, Watchdog, …
   • Initial configuration of RTE
Automated ECU Configuration: NvM Service Concept

Application developers define service needs → Nv Block Needs

Automated Configuration including Ports Connection

Automated Configuration

Application Component

NvM Service Component

NvM Ecuc Module

Refers to

NvM Standardized AUTOSAR Interfaces (Library)
Automated ECU Configuration: ‘Nv Block Needs’ UI
6. Summary

- Tool Implementation supporting AUTOSAR Methodology
  - As ONE Tool

- Some Customizations of AUTOSAR Methodology
  - CAN DBC file to Software Components
  - Framework of ECU Configuration Generation
  - More Automation: Platform SRS
  - Stack-wise ECUC Editor

- Integrated Development Environment for Automotive Software Developers
  - Developers stays in ECLIPSE most of the time.