Eclipse-based Embedded Engineering Environments Inspired by AUTOSAR

Stephan Eberle, Geensys

About...

• Stephan Eberle
  • Paris, France
  • Development lead of Geensys' AUTOSAR Builder product
  • Committer for EMFT Teneo component
  • 10+ years experience in OO and automotive embedded
  • Frequent presenter at conferences and events

• Geensys
  • Global Embedded Electronics & Networked SYstem Solutions
  • Embedded system engineering and IP
  • Embedded system engineering tools
  • Consulting services
Outline

- AUTOSAR from 4000 meters
- Need for custom AUTOSAR tools
- Definition of an open AUTOSAR tool platform
- Conclusion

Objectives of AUTOSAR

= AUTomotive Open System ARchitecture

- Mastering the ever increasing number of functions and complexity of E/E systems
  - Getting out of “one function – one ECU” trap
  - Transparent collaboration between OEMs and suppliers and reduction of integration pain
  - Reuse of E/E functions
    - In different product lines
    - With different hardware platforms/communication systems
How it works

- Decoupling of application software from ECU hardware and communication systems through VFB/RTE
- Formal (XML-based) description of
  - Application software components
  - ECU hardware
  - System topology
- Arbitrary mapping of application software components to ECUs
- Formal (XML-based) configuration of standardized basic software modules
  - Operating system
  - Device drivers
  - Communication drivers

Geensys’ Eclipse-based AUTOSAR Tooling

- AUTOSAR Builder
  - AUTOSAR Authoring Tool (AAT)
  - ECU Extract (EEX)
  - Generic Configuration Editor (GCE)
  - AUTOSAR Simulator (ASIM)
- SW-C Conformance Validation Tool (SCVT)
- Runtime Environment Generator (RTEG)
- Configuration Code Generator (CCG)
- Eclipse RCP/Plug-in SDK
To do list for AUTOSAR adaptors

- Coping with migration issues, e.g.
  - Import/conversion of legacy description formats
  - Conversion between AUTOSAR releases
- Handling of domain/vendor-specific aspects within AUTOSAR, e.g.
  - Automatic configuration of BSW modules from ECU Extract
  - Restriction of AUTOSAR design activities wrt custom development process and roles
- Integration with non-AUTOSAR development tools, e.g.
  - Model-based design tools
  - Build tool chains

What does all that mean to AUTOSAR tooling?

- AUTOSAR tool environments
  - Will never be complete out of the box
  - Must be highly adaptable to domain/vendor-specific contexts
- Idea
  - Provide an open and extensible AUTOSAR tool platform as common core for off-the-shelf and custom tool components
  - Final environment must enable both creating and using AUTOSAR tool components

\(\text{Solution:}\)

- Extend Eclipse Platform towards an AUTOSAR tool platform
Off-the-Shelf AUTOSAR Tools
- AAT
- EEX
- GCE
- SCVT
- RTEG
- CCG
- ASIM

Custom AUTOSAR Tools
- Legacy to AUTOSAR converters
- MBD Bridges
- Legacy-aware RTE Generators
- Configuration Assistants

AUTOSAR Tool Development Kit (TDK)

Eclipse RCP/Plug-in SDK

AUTOSAR TDK Components

AUTOSAR TDK
- AUTOSAR & BSW meta models/ APIs
- Basic AUTOSAR validation rules
- Basic AUTOSAR UI parts:
  - Explorer view
  - Properties sheets
  - Element wizards
  - Tree editor
  - Form editor
- Common AUTOSAR facilities:
  - Element search
  - Code generation interface
- Basic AUTOSAR project & preferences management

Eclipse
- IDE Workbench
- SWT, JFace
- Common Navigator
- EMF, EMF.Edit
- EMF Search
- EMF Validation
- GMF
- CDT
- Acceleo, Xpand
- Core Resources
Conclusion

• AUTOSAR is used in many different domain/vendor-specific contexts
• It is unlikely that a single AUTOSAR tool can satisfy all needs
• Providing an open and extensible tool platform is crucial for achieving the required degrees of customizability