Automotive System and Software Architecture

Yanja Dajsuren

2IW80 Software specification and architecture
March 25, 2014

TU/e
Technische Universiteit Eindhoven
University of Technology

Where innovation starts
Which one has more software?

Boeing 787, the most electronic airliner
~8,000,000 LOC

Chevrolet Volt, an example modern day car
~40,000,000 LOC
Why more software?

Enabling innovation easier and cheaper

Faster time to market

Decreasing development cost
Automotive supply chain software integration

**Architecture-driven:**
- (Partially) Automated
- Early detection of errors
- Less effort/cost to change

**Document-centric:**
- Manual
- Error prone
- Costly to change

Adapted from http://www.edibasics.hu/edi-resources/edi-by-industry/automotive.htm
Automotive architecture modeling

- Top-down system development i.o. bottom up
- Separation of concerns in different architectural models/views
- Model-driven i.o. document-centric approach
- Improved design quality by detecting errors early
- ...
Automotive companies and ADLs

- Automotive Modeling Language (AML)
- COmponent Language (COLA)
- EAST-ADL
- Timing Augmented Description Language (TADL)
- The ICT MAENAD project EAST-ADL2
EAST-ADL

- Advancing Traffic Efficiency and Safety through Software Technology 2 (ATESST) project
- Refined EAST-ADL2 language, profile, methodology, tools
- It provides means to represent the embedded system in several abstraction levels.
- Main source: http://www.east-adl.info/
EAST-ADL and AUTOSAR

Features of the vehicle
Abstract functions
Hardware topology, concrete functions, allocation to nodes
Software Architecture as represented by AUTOSAR

http://maenad.eu/
EAST-ADL Abstraction Levels

TechnicalFeatureModel

ExampleFeatureTree

DoorLock

BaseBrake

ABS

BrakeLight

Realization relations

Lock Button

Lock Request

Wheel Speed Sensor

Wheel Speed

Brake Pedal

PedalBrk Request

Vehicle SpeedCalc

Vehicle Speed

Brake Controller

Brake Request

Lock Controller

Lock Activation

Brake WheelCtrl

Brake Force

Lock Actuator

Brake Actuator

FunctionalAnalysisArchitecture

Analysis Level

Vehicle Level
EAST-ADL Abstraction Levels

System Model

Design Level
- FunctionalDesignArchitecture
  - Functions
  - MW
  - Sensors/Actuators
- HardwareDesignArchitecture

Implementation Level
- AUTOSAR SWC Template
- AUTOSAR ECU Resource Template
- AUTOSAR System Template

Environment Model
Example of function-to-component Mapping

n Function to 1 SW Component
EAST-ADL Metamodel Structure

Vehicle Level
- VehicleLevel
  - TechnicalFeatureModel

Analysis Level
- AnalysisLevel
  - FunctionalAnalysisArchitecture

Design Level
- DesignLevel
  - FunctionalDesignArchitecture
  - HardwareDesignArchitecture

Implementation Level
- ImplementationLevel
  - AUTOSAR Application SW
  - AUTOSAR Basic SW
  - AUTOSAR HW

Data exchange over ports → Allocation

Extensions ...
- Requirements
- Variability
- Timing
- Dependability

TU/e
Technische Universiteit Eindhoven
University of Technology
Traceability between abstraction levels

Realization relations identify which abstract element is realized by a more concrete entity.

- Functions on analysis level realizes features on vehicle level.
- Functions on design level realizes functions on analysis level.
- SW components or runnables on implementation level realizes functions on design level.
EAST-ADL Tooling

UML-based Tooling
- CEA Papyrus
- Integrated Eclipse application with 5 ATESSST plugins
- MagicDraw UML
- ...

DSL AUTOSAR Tooling
- MentorGraphics VSA

DSL Generic Tooling
- MetaEdit+
- TopCased
EAST-ADL Summary

- Defines several abstraction levels and mapping between them
- Extensions to traditional ADLs:
  - Requirements
  - Variability
  - Timing
  - Dependability
  - Safety (alignment with ISO26262)
  - Environment modeling
- Not well applied yet in automotive industry
SysML and UML

- Not required by SysML
- SysML’s extensions to UML
- UML reused by SysML (UML4SysML)
SysML Diagram Taxonomy

- Activity Diagram
- Sequence Diagram
- State Machine Diagram
- Use Case Diagram
- Block Definition Diagram
- Internal Block Diagram
- Package Diagram

- Same as UML 2
- Modified from UML 2
- New diagram type
Blocks are Basic Structural Elements

• Provides a unifying concept to describe the structure of an element or system
  • System
  • Hardware
  • Software
  • Data
  • Procedure
  • Facility
  • Person

• Multiple standard compartments can describe the block characteristics
  • Properties (parts, references, values, ports)
  • Operations
  • Constraints
  • Allocations from/to other model elements (e.g. activities)
  • Requirements the block satisfies
  • User defined compartments

| Compartment Label | values | \( allocatedFrom \)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{DutyCycle} ): Percentage</td>
<td>«activity» Modulate BrakingForce</td>
<td>«block» BrakeModulator</td>
</tr>
</tbody>
</table>
Using Blocks

• Based on UML Class from UML Composite Structure
  • Supports unique features (e.g., flow ports, value properties)
• Block definition diagram describes the relationship among blocks (e.g., composition, association, specialization)
• Internal block diagram describes the internal structure of a block in terms of its properties and connectors
• Behavior can be allocated to blocks
Block Definition vs. Usage

**Definition**
- Block is a definition/type
- Captures properties, etc.
- Reused in multiple contexts

**Usage**
- Part is the usage of a block in the context of a composing block
- Also known as a role
Internal Block Diagram (ibd) Specifies Interconnection of Parts

Enclosing Block

Connector

Item Flow

Port

Part
Reference Property Explained

• S1 is a reference part*
• Shown in dashed outline box

*Actual name is reference property
SysML Ports

- Specifies interaction points on blocks and parts
  - Integrates behavior with structure
  - portName:TypeName

- Kinds of ports
  - Standard (UML) Port
    - Specifies a set of required or provided operations and/or signals
    - Typed by a UML interface
  - Flow Port
    - Specifies what can flow in or out of block/part
    - Typed by a block, value type, or flow specification
    - Atomic, non-atomic, and conjugate variations

Standard Port and Flow Port Support Different Interface Concepts
Port Notation

**Standard Port**

- **provided interface** (provides the operations)
- **required interface** (calls the operations)

**Flow Port**

- **item flow**
State Machines

- Typically used to represent the life cycle of a block
- Support event-based behavior (generally asynchronous)
  - Transition with trigger, guard, action
  - State with entry, exit, and do-activity
  - Can include nested sequential or concurrent states
  - Can send/receive signals to communicate between blocks during state transitions, etc.
- Event types
  - Change event
  - Time event
  - Signal event
Operational States (Drive)

Transition notation: trigger[guard]/action
Adaptive Cruise Control (ACC) in SysML

Modeling the ACC system for an E-truck with a top-down approach in SysML

Image: http://www.extremetech.com/
Requirements Diagram

Source: Artisan Software Tools
Use Case diagram

- Provides means for describing basic functionality in terms of usages of system by actors
- Generally elaborated via other behavioral representations to describe detailed scenarios

Source: Artisan Software Tools
System architecture
System integration

- Software
- Hardware
Running ACC_UI on Freescale board

- **Test interface** (windows PC)
- **Serial interface**
- **Freescale board**
SysML summary

- SysML provides a general purpose modeling language to support specification, analysis, design and verification of complex systems
  - Subset of UML 2 with extensions
  - 4 Pillars of SysML include modeling of requirements, behavior, structure, and parametrics
- Intended to improve communications, tool interoperability, and design quality
- Multiple tools available
  - IBM –Rhapsody
  - Sparx Systems -Enterprise Architect
  - Atego –Artisan Studio etc.
Automotive supply chain software integration

- Hardware dependent SW
- Not efficient software reuse and exchange
- Costly integration

Adapted from http://www.edibasics.hu/edi-resources/edi-by-industry/automotive.htm
AUTOSAR (AUTomotive Open System Architecture)

- An open and standardized automotive software architecture

- Architecture
- Methodology
- Application Interfaces
AUTOSAR Milestones

- 2003: AUTOSAR founded
  - First release
- 2005: Basic SW complete
  - Derived applications
- 2006: Feature enrichment
  - New development methods
- 2009: Release 4.1.1
  - Multicore support
  - Functional safety
  - Ethernet
  - ...
AUTOSAR Layered Architecture

http://autosar.org/
AUTOSAR Methodology

Virtual Integration
Virtual Functional Bus - Independent of hardware

Introduction of HW Attributes
Holistic view of the entire system, both software and hardware

ECU Configuration
Separation of system into its ECUs with a common SW platform (infrastructure).

http://autosar.org/
AUTOSAR Application Interface

Syntax of Interfaces:
- Meta-model, Software Component Template
- Supporting transferability within the network

Semantics of Interfaces:
- Physical properties, units, etc.
- Supporting re-use across product lines
- In scope of AUTOSAR workpackages specifying application interfaces

http://autosar.org/
AUTOSAR Use Case

Use case ‘Front-Light Management’ in AUTOSAR

http://autosar.org/
AUTOSAR Benefits

Scenario A
The supplier provides the ECU to a different OEM.

Scenario B
Integration of features, delivered from different sources.

Scenario C
The hardware changes.

http://autosar.org/
Automotive Standards

- **ISO 26262:**
  - Absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems

- **IEC 61508:**
  - Part of the overall safety related to the equipment under control (EUC) that depends on the correct functioning of the safety-related system.

- **MISRA C:**
  - Software development standard
ISO 26262

IEC 61508
Functional Safety for E/E/PES Safety Related Systems

IEC 61511
Process Industry

IEC 62061
Machinery

IEC 61513
Nuclear

ISO 13849-1
Machine Safety

ISO 26262
Road Vehicles

ISO 25119
Tractors...

ISO 26262 is “State of the Art” For Automotive Developed with OEM

KoenLeekens, ISO-26262 introduction, 2012
Safety in V cycle
Safety Analysis in ISO 26262

1. Vocabulary

2. Management of functional safety
   - 2.6 Overall safety management
   - 2.7 Safety management during the concept phase and the product development
   - 2.8 Safety management after the item’s release for production

3. Concept phase
   - 3.5 Item definition
   - 3.6 Initiation of the safety lifecycle
   - 3.7 Concept

4. Product development at the system level
   - 4.6 Specification of the technical safety requirements
   - 4.9 Item integration and testing
   - 4.11 Release for production
   - 4.10 Functional safety assessment

5. Product development at the hardware level
   - 5.4 Hazard analysis
   - 5.5 Hazard analysis and risk assessment
   - 5.6 Failure mode and effect analysis

6. Product development at the software level
   - 6.4 Hazard analysis
   - 6.5 Hazard analysis and risk assessment
   - 6.6 Failure mode and effect analysis

7. Production and operation
   - 7.5 Production
   - 7.6 Operation, service (maintenance and repair), and decommissioning

8. Supporting processes
   - 8.5 Interfaces within distributed developments
   - 8.6 Specification and management of safety requirements
   - 8.7 Configuration management
   - 8.8 Change management
   - 8.9 Verification
   - 8.10 Documentation
   - 8.11 Confidence in the use of software and hardware
   - 8.12 Qualification of software component
   - 8.13 Qualification of hardware component
   - 8.14 Proven in use argument

9. ASIL-oriented and safety-oriented analyses
   - 9.5 Requirements decomposition with respect to ASIL tailoring
   - 9.7 Analysis of dependent failures

10. Guideline on ISO 26262
    - 9.6 Criteria for coexistence of elements
    - 9.8 Safety analyses

H&R: Hazard & Risk
SCA: System Criticality
FTA: Fault Tree
FMEA: Failure Mode Effect
FMEDA: FMEA with Diagnostics
SWCA: SW-Criticality
HAZAN: Hazard Analysis

KoenLeekens, ISO-26262 introduction, 2012
MISRA C

- MISRA C is a software development standard for the C programming language developed by MISRA (Motor Industry Software Reliability Association).
- Its aims are to facilitate code safety, portability and reliability in the context of embedded systems, specifically those systems programmed in ISO C.
- As with many standards the MISRA C guideline documents are not free to users or developers.
Summary

- In the automotive industry, more and more software and electronics system require system and software architecture methods.

- Automotive specific and generic purpose ADLs are being developed and applied.

- Many stakeholders, functionalities, safety and environment requirements require automotive specific standards.
Contact for comments and questions:

Tel: +31(0)402475052
Email: y.dajasuren@tue.nl
Address:
  MF 7.123, Eindhoven University of Technology
  5612 AZ Eindhoven, The Netherlands
Automotive Modeling Exercise

Design a **PowerWindow System** of a vehicle:

A. Elicit requirements

B. PowerWindow system architecture (decompose the system into software and hardware components)
Required Software Tools

IBM Rational Rhapsody 8.0.5

Matlab R2011b