“Advancing Traffic Efficiency and Safety through Software Technology”

Managing Complexity of Automotive Electronics Using the EAST-ADL

UML & AADL Workshop, July 14, 2007

Outline

Part I. Overview of ATESTT Project
Part II. Overview of EAST-ADL
Part III. Variability Management in EAST-ADL2
Part I
Overview of ATESST Project
What is ATESST?

Advancing Traffic Efficiency and Safety through Software Technology
in EU 6th Framework Programme, DG Information Society Technologies
Specific targeted research project, STREP
2006 – 2008 Q1

Scope:

• Model-based development for embedded automotive systems
• Definition of an architecture description language and methodology
ATESST Partners

Vehicle Manufacturers
- DaimlerChrysler
- Volkswagen/Carmeq
- Volvo Cars
- Volvo Technology (coordinator)

Suppliers/Tool Vendors
- ETAS
- Mecel AB
- MentorGraphics
- SiemensVDO

Academic
- CEA
- KTH
- TU Berlin
History

1999/2001

ITEA EAST-EEA

2005

WP3

ADL

2006

AUTOSAR

SystemML

UML SPT/

Marte

2007

WP x

AUTOSAR

ADL

„EAST-ADL 2“

2008

ATESST
Objective and Outcome

Objective of ATESST

Definition of a comprehensive, standardized ADL for the automotive domain based on the existing approach EAST ADL, including …

• Software & Hardware Components, Communication
• Environment modelling
• Requirements and V&V
• Variant handling and product families
• …

Outcome of ATESST

• Architecture Description Language (Revised EAST ADL as UML2 profile)
• Documentation and methodology
• Prototype tool
• eSafety demonstrator
EAST-ADL Motivation

Product Aspects
- Functionality increase
- Complexity increase
- Increased safety-criticality
- Increased infra-structure complexity
- Increased coupling between vehicle functions and vehicle-to-vehicle interaction
- Quality concerns

Development Aspects
- Supplier-OEM relationship
- Multiple sites & departments
- Product families
- Separation of application from infrastructure / function orientation
Re-inventing The Wheel?

Why Not UML?
• ATESTST works with a specialization of UML2

Why not SysML?
• ATESTST is a specialization of applicable SysML concepts

Why not Autosar?
• ATESTST Complements Autosar with VFM, functional architecture (abstract and concrete), requirements, V&V, variability management, environment modeling

Why not proven proprietary tools (Simulink, Statemate, …)
• ATESTST integrates external tools and provides an information structure for the engineering data regardless of tool
ATESST and AUTOSAR

“AUTOSAR prescribes everything that is needed to allow several AUTOSAR Software Components to be integrated correctly in an infrastructure consisting of networked ECUs.”
ATESST and AUTOSAR

ATESST Contribution

• What were the requirements?

• What are the user level features (vehicle configuration)?

• What about the abstract functional content? How are the SW components’ functions related?

Conclusion:
AUTOSAR concepts are integrated on implementation level and down
ATESST Tool Prototype

Eclipse based
  • Eclipse EMF/UML2

Built on Papyrus (UML Modeling Tool)

Open Source

Plug-ins planned for
  • Variability
  • Requirements
  • Analysis / Derivation
ATESST Demonstrator

Active Safety Application

Used in several steps

1. Simulink
2. EAST ADL / UML Tool (MagicDraw)
3. EAST ADL / ATESST Tool
4. Tool experiments (Code generation, variability, requirements)

Contains Engine control, ACC, CMbB (based on PReVENT application), Speed regulation, Environment Model
Part II
Overview of EAST-ADL2
Levels of Abstraction in EAST-ADL2

An Information Model
That captures engineering information in a standardized way

- Vehicle Level
- Analysis Level
- Design Level
- Implementation Level
- Operational Level

functional definition of software; independent of hardware topology

reusable platform-independent code and system configuration for hardware deployment

final binary software deployment
EAST ADL2 Structure

Vehicle Level
- Vehicle Feature Model

Analysis Level
- Analysis Architecture
  - Functional Analysis Architecture

Design Level
- Design Architecture
  - Functional Design Architecture
  - Basic SW Architecture

Implementation Level
- Implementation Architecture
  - Application SW Architecture
  - Hardware Architecture

Operational Level
- Operational Architecture
AUTOSAR Integration

- AUTOSAR entities populate architectures on Implementation Level
- Requirements, variability constructs, traceability relations link to AUTOSAR entities
- Behavioral semantics adapted to map to AUTOSAR behavioral concept
SysML Harmonization

- EAST ADLFunction on analysis and design level inherits from SysML block
- Port and datatype concepts from SysML used
- Requirements constructs from SysML used
- Parametric diagrams from SysML potential candidate for certain plant models
Behavior Modeling

• External and Native Behavior are investigated
  External via URL and defined execution semantics
  Native behavior preliminarily from UML & SysML

• Compatible with state of practice tools Simulink, ASCET...
  Relation with URL
  Code generated from this part of the external model is the implementation code and referenced in model

• Harmonization with AUTOSAR via relation to runnable entities

• Ongoing investigation how environment models should be described in EAST-ADL2.0
Requirements

Behavioral Models that specify the requirement in more detail

System Components which have to satisfy the requirement

New requirements derived after a system decomposition or system refinement

Verify-Req

V&V-Cases that verify the requirement

Supporting Requirements Tracing and Linking to system components through EAST-ADL viewpoint (from VFM to Implementation)
Others Requirements Modeling

- Hazards or hazardous events defined modeled in the environment model

- Safety requirement attributes
  Relation to hazards event
  Include safety integrity level (SIL), operation state, fault time span, emergency operation times, safety state, and functional redundancy.

- Timing requirement attributes
  Minimum, maximum, Jitter for concept such delays, synchronisation point and interval

- V&V Case modeling
  combination of a V&V-case, its environment and its target object is described as a V&V context
Error Modeling

Extending EAST-ADL with error modeling for enabling the integration of architecture design and safety analysis in an efficient way.
Part III
Variability Management

1. Overview
2. Basic Concepts
3. Advanced Concepts
Variability – Overview

Variability is modeled essentially on two different abstraction layers:

1. Feature layer variability (being the primary source for variant/product configuration)

2. Artifact layer variability (comprising all artifact elements, e.g. requirements, FAA, FDA…)

Diagram:

- Environment
  - Vehicle Level
  - Analysis Level
  - Design Level
  - Implementation Level
  - Operational Level

Feature layer variability configures Artifact layer variability
Basic Concepts

- Vehicle Level
- Analysis Level
- Design Level
- Implementation Level
- Operational Level

Feature Modeling

Variation Points
Basic Concepts – Feature Modeling

• as introduced by Kang et al. in 1990
What is a Feature?

A Feature is a characteristic or trait that the variants of a variable entity may or may not have.
What is a Feature?

- a Feature can represent things of very different nature

Specializations
- Wiper
  - Basic
  - Advanced
- Car
  - Engine Control
  - Infotainment / Telematics
  - Body Electronic System
- Subsystem / Subdomain
  - SpeedLimit : Int

Functionality
- LowEnergy Consumption
  - Speed-Ctrld
  - Rain-Ctrld
- CruiseControl
  - Simple
  - Adaptive
  - Actuator
- Radar
- Hardware-Component

/feature2right

A Feature can represent things of very different nature
Basic Concepts – Variation Points

Variability is modeled for the artifacts based on the notion of variation points and variation groups:

- all ADLEntities can be variable
- variable ADLEntities are marked a variation point (and then serve as a placeholder)
- the variation point is linked to the entities that represent the available variants
- variation groups can describe various constraints between arbitrary variable entities
Advanced Concepts

View concept for End-Customer Configuration (and others)

Model Range Spanning Variability

Hierarchical Composition of Variability

Environment

- Vehicle Level
- Analysis Level
- Design Level
- Implementation Level
- Operational Level

EE Architecture
End-Customer Configuration

- vehicle level contains the core feature model
  - showing variability of the complete system
  - high complexity
  - technical viewpoint
    (terminology, customer-invisible variability, diverse life-cycle)
  - not appropriate for end-customer configuration

- vehicle level supports to define end-customer configuration
  - provides „view“ on core feature model
  - allows for orthogonal „packaging“ of variability
  - supports orthogonal configuration considerations
  - can be used for separation of other concerns
Model Range Spanning Variability

two kinds of variability:

• model range specific variability
• model range spanning variability
Model Range Spanning Variability

- traditional solution: either one large software product line or several independent ones
- multi-level concept: compromise between the two
  - sublines can be managed locally
  - concordance can still be managed globally
A has variability inside.

When A is used twice in B, then it must be configured separately for each occurrence.

→ Thus, definition of A is not a suitable context for configuration!!
Composite Variability

- variability is managed in a hierarchical manner

- key elements:
  - Step 1: ADLFunctons get a „public“ feature model
  - Step 2: internal structure of composite ADLFunctons can be variable (by way of variation points)
  - Step 2: for each composite ADLFuncton a mapping is defined from its public feature model to …
    (a) the public feature models of its contained ADLFunctons and
    (b) the variants of the contained variation points

(in other words: it is defined how the configuration of the contained ADLFunctons' feature models can be derived from a given configuration of the public feature model of the container ADLFuncton)
Step 1: ADLFunctions get „public“ FM

• as part of their public interface

• for elementary ADLFunctions, the configuration of the public feature model will be made available for reference within the behavioral description of the ADLFunction
  • details depend on type of behavioral description
  • e.g. for C-Source-Code, parameters can be realized as pre-processor macros

EXAMPLE: ElemSWComp that realizes a „Threshold“

```
    input :Float
    output :Boolean

    thresholdValue:Float

    if input >= thresholdValue
        then output = True
    else output = False.
```

```
EXAMPLE: CompSWFunc that controls a wiper depending on driver input and rain sensor
Step 3: Mapping

- mapping from the public feature model of a composite ADLFunction to
  (a) the public feature models of its contained ADLFunctions and
  (b) the variants of the contained variation points
- in this case: only (a) used

NOTE:
Here thresholdValue is simply propagated up the containment hierarchy

BINDING-SCENARIO #1
Applying WiperControl

NOTE:
Here rainThreshold is directly and invariantly(!) bound instead of being propagated up

BINDING-SCENARIO #2
Alternative

WiperControl

thresholdValue :=
front ? 140 :
rear ? 80 : undef

NOTE:
Here, two kinds of bindings of
thresholdValue are defined and selection between the two
is propagated up
BINDING-SCENARIO #3
Feature Models in ATESSST

VFM

Customer-Config Feature Model

Vehicle Feature Model (= the core feature model)

FDA

public FM of ADLFunction Car (= top-level ADLFunction)

public FM of ADLFunction ClimateSystem

public FM of ADLFunction RainSensor

public FM of ADLFunction WiperSystem

public FM of ADLFunction WiperControl
Summary

ATESST Project defines EAST-ADL2

- refinement of EAST-ADL
- integration of Autosar and other recent standards
- contribution of concepts to existing efforts (OMG, Autosar, Proprietary)

includes

- Software & Hardware Components, Communication
- Environment modelling
- Requirements and V&V
- Variant handling and product families

- variant handling based on feature modeling
- supports end-customer configuration, model range spanning variability, hierarchical composition of variable entities