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

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

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

FunctionalAnalysisArchitecture

LockButton

LockRequest

WheelSpeedSensor

WheelSpeed

BrakePedal

PedalBrkRequest

VehicleSpeedCalc

VehicleSpeed

BrakeController

BrakeRequest

BrakeWheelCtrl

BrakeForce

LockController

LockActivation

LockActuator

BrakeActuator
EAST-ADL Abstraction Levels

System Model

Design Level
- Functional Design Architecture
- Hardware Design Architecture

Implementation Level
- AUTOSAR SWC Template
- AUTOSAR ECU Resource Template
- AUTOSAR System Template
Example of function-to-component Mapping

n Function to 1 SW Component
EAST-ADL Metamodel Structure

SystemModel

- VehicleLevel
  - TechnicalFeatureModel

- AnalysisLevel
  - FunctionalAnalysisArchitecture

- DesignLevel
  - FunctionalDesignArchitecture
  - HardwareDesignArchitecture

- ImplementationLevel
  - AUTOSAR Application SW
  - AUTOSAR Basic SW
  - AUTOSAR HW

Extensions...

- Requirements
- Variability
- Timing
- Dependability

Data exchange over ports
Allocation
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

<table>
<thead>
<tr>
<th>«block»</th>
<th>BrakeModulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocatedFrom</td>
<td>«activity» Modulate BrakingForce</td>
</tr>
<tr>
<td>values</td>
<td>DutyCycle: Percentage</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

**Block Definition Diagram**

**Internal Block Diagram**

**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

Internal Block Diagram (ibd) Specifications

- Blocks, Parts, Ports, Connectors & Flows

Example Diagram:

- ibd [Block] Anti-Lock Controller [Item Flow]
  - c2:
  - d1: Traction Detector
  - activate: 5vdc
  - m1: Brake Modulator

These components illustrate the interconnection of parts within a system.
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)*

**Flow Port**

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

**item flow**

```
part1:  part2:

part1:  part2:
```
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
PAUSE
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

- **REQ_CCS_01**: The CCS must allow a driver to enable the vehicle to maintain a desired speed.
  - *Maintain Speed*
  - *Cruise Control System*

- **REQ_CCS_02**: The CCS must allow cruise control to be engaged and disengaged. When engaged, the cruise control system is available to accept driver instructions (such as 'set' and 'increment'). When disengaged, the cruise control system will not respond to any driver inputs.
  - *Engage CC*
  - *Disengage CC*

- **REQ_CCS_03**: The CCS must allow cruise control to be suspended (via toggle or brake application) and resumed (via toggle only). This feature should be available only when cruise control is active. When suspended, the cruise control system shall memorize the desired speed, but the cruise control system is inactive (i.e., suspended) and must relinquish control of the throttle pedal back to the driver. The only commands which can be accepted whilst suspended are 'resume', 'disengage' or 'stop'.
  - *Suspend CC*
  - *Resume CC*

- **REQ_CCS_04**: The failsafe state for the CCS is 'disengaged'; any errors encountered by the cruise control system shall be logged (along with the system's configuration data) and the system shall be disengaged.

- **REQ_CCS_05**: Once the CCS is engaged, to activate cruise control the driver can 'set' the desired speed. Once this is set the CCS shall take over control of the throttle.
  - *Set Desired Speed*

- **REQ_CCS_06a**: When cruise control is engaged, the driver must be able to increment the desired speed in increments of 1 MPH. The driver must also have the ability to change the gear selection whilst the cruise control is active.
  - *Increment Speed*

- **REQ_CCS_06b**: When cruise control is engaged, the driver must be able to decrement the desired speed in increments of 1 MPH.
  - *Decrement Speed*

- **REQ_CCS_06c**: When cruise control is active, the driver must be able to change the gear selection.
  - *Shift Gear*

- **REQ_CCS_07**: The CCS must provide displayed outputs to the driver. This will enable the driver to determine the current desired speed before resuming cruise control.
  - *Display Speed*
  - *Cruise Control System - block operation*

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
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 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).
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

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

1. Vocabulary

2. Management of functional safety

2.6 Safety management during the concept phase and the product development

2.7 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 Item integration and testing

3.8 Hazard & Risk

3.9 FMEA

3.10 FTA

3.11 HAZAN

4. Product development at the system level

4.6 Specification of the technical safety requirements

4.7 Safety management during the concept phase and the product development

4.8 Safety management after the item’s release for production

4.9 Item integration and testing

5. Product development at the hardware level

5.1 Design of the hardware and its environment

5.2 FMEA

5.3 FMECA

6. Product development at the software level

6.5 Specification of the software safety requirements

6.6 Qualification of software components

6.7 Software architectural design

6.8 Software unit design and implementation

6.9 Software integration and testing

6.10 Verification of software safety requirements

7. Production and operation

7.5 Production

7.6 Operation, service, repair, and decommissioning

8. Supporting processes

8.6 Specification and management of safety requirements

8.7 Configuration management

8.8 Change management

8.9 Verification

9. ASIL-oriented and safety-oriented analyses

9.5 Requirements decomposition with respect to ASIL tailoring

9.7 Analysis of dependent failures

9.8 Safety analyses

9.9 Criteria for coexistence of elements

10. Guideline on ISO 26262

H&R: Hazard & Risk
SCA: System Criticality
FTA: Fault Tree
FMEA: Failure Mode Effect
FMECA: FMEA with Diagnostics
SWCA: SW-Criticality
HAZAN: Hazard Analysis
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.
2014 record recall year

Software problem that could cause

- the cars to **stop suddenly**
- accelerate **without warning**
- **overheats/damages** power electronics
- ...

<table>
<thead>
<tr>
<th>YEAR</th>
<th>TOTAL RECALLS ISSUED</th>
<th>TOTAL NO. OF VEHICLES AND EQUIPMENT RECALLED IN MILLIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td>269</td>
<td>18.5</td>
</tr>
<tr>
<td>1991</td>
<td>282</td>
<td>14.4</td>
</tr>
<tr>
<td>1992</td>
<td>217</td>
<td>13.6</td>
</tr>
<tr>
<td>1993</td>
<td>264</td>
<td>11</td>
</tr>
<tr>
<td>1994</td>
<td>290</td>
<td>9.9</td>
</tr>
<tr>
<td>1995</td>
<td>348</td>
<td>19</td>
</tr>
<tr>
<td>1996</td>
<td>341</td>
<td>19.5</td>
</tr>
<tr>
<td>1997</td>
<td>312</td>
<td>16.7</td>
</tr>
<tr>
<td>1998</td>
<td>408</td>
<td>19.2</td>
</tr>
<tr>
<td>1999</td>
<td>449</td>
<td>55.6</td>
</tr>
<tr>
<td>2000</td>
<td>626</td>
<td>44.6</td>
</tr>
<tr>
<td>2001</td>
<td>627</td>
<td>22.4</td>
</tr>
<tr>
<td>2002</td>
<td>506</td>
<td>25.3</td>
</tr>
<tr>
<td>2003</td>
<td>600</td>
<td>22.9</td>
</tr>
<tr>
<td>2004</td>
<td>698</td>
<td>33</td>
</tr>
<tr>
<td>2005</td>
<td>645</td>
<td>20.4</td>
</tr>
<tr>
<td>2006</td>
<td>613</td>
<td>14.1</td>
</tr>
<tr>
<td>2007</td>
<td>713</td>
<td>20.6</td>
</tr>
<tr>
<td>2008</td>
<td>781</td>
<td>22.6</td>
</tr>
<tr>
<td>2009</td>
<td>571</td>
<td>18</td>
</tr>
<tr>
<td>2010</td>
<td>723</td>
<td>23</td>
</tr>
<tr>
<td>2011</td>
<td>657</td>
<td>17.5</td>
</tr>
<tr>
<td>2012</td>
<td>657</td>
<td>18.1</td>
</tr>
<tr>
<td>2013</td>
<td>714</td>
<td>27</td>
</tr>
<tr>
<td>2014</td>
<td>500</td>
<td><strong>56</strong></td>
</tr>
</tbody>
</table>

Source: National Highway Traffic Safety Administration
...And I’m saving even more on gas now that I’m afraid to drive it.
Example applications

- Functional domains
  - Power Train
  - Chassis-Safety
  - Body Electronics
  - Infotainment

- Example applications
  - Engine
  - Transmission
  - Braking, Steering
  - Airbag
  - Door Modules, Anti-theft
  - Lighting, Wipers
  - HVAC, Cluster
  - Telematics, GPS
  - Car Infotainment
  - PND

Hybrid Innovations for Trucks (HIT) project

Safety-Critical Domain Certification

InMotion, Solar Team, “Cars in Context” TU/e projects

**OPENCOSS**
HIT Results

- Architecture modeling approaches
  - ADL evaluation
  - Architecture framework

- Quality framework
  - Validated metrics for Simulink
  - Metrics tool
  - Visualization
Energy Management (EM) Evaluation

- Integrated Energy and Thermal Management (IETM) model
- Integrated Energy and Emission Management (IEEM) model
EM Evaluation

- Model clones may have the effect of increasing code size and duplication of errors.
- Number of duplicates affects modifiability as well.
Contribution to GMM based Editors

Generic MetaModel

Metamodel Transformation

1

Specific MM

SM Editor

Add domain concepts

Automatic generated

Link to

Safety Case

Domain Concepts from Standard (ISO 26262) or Project

Structured English

TU/e Technische Universiteit Eindhoven University of Technology
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:

Yanja Dajsuren
Tel: +31(0)402475052
Email: y.dajsuren@tue.nl
Address:
MF 6.085, Eindhoven University of Technology
5612 AZ Eindhoven, The Netherlands