Model-based Development in the Upcoming Automotive Embedded Software Architecture of AUTOSAR

by Bruce D. Emaus

President of Vector CANtech & SAE Embedded Software Standards Committee Chair

Automotive Software Workshop – San Diego
March 2006

All opinions expressed in this presentation are based solely on the business and technical experiences of Bruce Emaus.

Bruce Emaus - Biography

- President of Vector CANtech, a company that specializes in development tools and embedded software components for vehicle-based distributed applications.
- 30 years of automotive experience covering embedded software, electronics hardware, information engineering, and systems engineering
- Expert in the area of electronic product development, distributed embedded systems, and small area network protocols
- Chairman of the SAE Embedded Software Standards Committee and co-chair of the SAE Distributed Embedded Systems Engineering Task Force
Vector – The company Bruce Emaus works for

- Vector is a global company located in Stuttgart, Germany
  - Subsidiaries in USA, Japan, France, Sweden
- Market leader - Small Area Network tools – CANalyzer, CANoe, and CANape
  - Protocols include: CAN, LIN, MOST, FlexRay, ...
- Market leader - In-vehicle Embedded Software Components
  - Software stacks include: Ford FNOS, GM LAN, DC NET, and others

SAE Embedded Software standardization efforts

- J2632 "C Coding Practices"
- J2640 "General Automotive Embedded Software Design Requirements"
- J2516 "Embedded Software Development Lifecycle"
- J2746 Embedded Software Development Process Assessments
- J2734 "Embedded Software Testing, Verification, and Validation"
- J2356 Embedded Software/Systems
- J2720 Embedded Software Development for Calibration & Manufacturing

*Just Beginning – April 13th Open Discussion – Model-based Development and Autocode Generation – Is There a Need for Automotive Industry Standardization?*

Any interest? Visit the SAE website OR add your name to the SAE Email list for these activities - so you can be aware of these activities...
- email <bruce.emaus@vector-cantech.com>
Overview

- As automotive product architectures continue to migrate toward higher levels of distributed-ness with increasing system and software complexity, the increased demand for model-driven automotive embedded software development continues to grow.
- The industry push for AUTOSAR - a new automotive software architecture - will impact modeling.

This presentation discusses:
- The essential business case for AUTOSAR.
- The design challenges of model-based software development in the automotive distributed embedded system domain.

Automotive Embedded Software – Enormous in Size

- Approximately ten years ago, the total amount of embedded software resident in one high-end OEM vehicle passed the level of 1 million lines of code and powertrain software was a significant portion.

- Today’s high-end luxury car is close to 100 M bytes of software (source: Hansen Report)
  - ~10M to 30 M lines of code.
- Major contributing areas to this increase:
  - More and more feature content
  - More powertrain requirements
  - Telematics
  - New stability control subsystems.
Automotive Embedded Software Growth Factors

- Increase in software implementation choices
- Increase in customer feature content
- Increase in government regulatory requirements
- Increase in the use of vehicle networking (multiplexing)
- Increase in the number of communication protocols
- Increase in the number of OEM-specific network operating strategies
- Increase in system architecture variation especially with the use of x-by-wire sub-systems

Too Many Software Implementation Choices

- Increasing number of different microcontrollers (including all derivatives)
- Increasing number of different programming languages (including all provider derivations)
- Increasing number of different modeling tools (including all model settings)
- Increasing number of different compilers (including all compiler settings)
- Increasing number of different debug tools (including all tool settings)
- Increasing number of different programmers (including all levels of skill – from the super cowboy to the mediocre)
- Increasing number of different software development processes (for all levels of assessment)
Impact of Increasing Feature Content

Example – Powertrain

- Implementing more powertrain features requires more I/O, increased packaging, and the use of partitioning plus the use of data communications
- All contribute to an increase in software

Impact of Government Regulatory Requirements

- Implementation of On Board Diagnostics (OBD) consumes up to half of the embedded software in a (gasoline-based) engine controller
- Add new regulatory requirements continue to grow - each new regulatory requirement increases the complexity and increases software
- Potential upcoming x-by-wire technologies will likely include new regulatory requirements
Managing software growth with model-based development

For automotive product creation, model-based development promises
- Business value AND Technical value

The value of modeling influences its adoption
1. The ability of the model to describe the highest level of the product behavior
   the higher the value to the enterprise
2. The more uses for the model across the development cycle the higher the
   value to the enterprise

Is model-based development currently at the highest value to the automotive
enterprise?

Current modeling issues

- Early models were used to describe a single-box product – now we have
  multiple boxes inside a distributed product architecture
- Original value was typically limited to a single engineer – now we would like
  the value to be usable by more than one engineer
- Early models just described feature operation – now models can be used to
  automatically generate embedded software
- Early modeling tools required substantial effort to learn – today’s modeling
  tools still take substantial effort to learn
- Early modeling tool choices were limited – now we have many modeling
  choices – do we have to many choices?
- Early modeling tools were used without attention to the development process
  – at some companies the modeling effort needs to support defined
    engineering processes
- Early models paid little attention to testing – now models can be also used to
  provide testing - future models are being asked to address automated
  testing
Enterprise-wide model-based development

- Virtually all modeling methods have technical value – but this value may be limited to one or just a few engineers.
- However - Once modeling extends across multiple engineering development processes or across multiple departments AND
- Once modeling can be used across the entire enterprise – including both the OEM and the OEM’s module suppliers,
- Then the overall value of modeling should increase.

Today - Enterprise-wide modeling is what several automotive OEMs are focusing on – in order to interconnect many engineering activities both internal and external (across the supplier base).
- Let’s examine the influences inside the distributed product development process...

Product architecture heavily influences modeling

- As an alternative to the “centralized” product approach, the Distributed Embedded System or distributed product architecture uses a partitioned and communication-based product architecture to achieve the same functionality with the potential advantages of lower cost, flexibility, and scalability.

![Centralized vs. Distributed Architecture Diagram](image-url)
The product architecture choice - centralized or distributed

- Based on business reasons, the automotive OEM has chosen to use a distributed product architecture to manage the large amount of in-vehicle complexity.
- Additionally, the automotive OEMs have also moved to a "supplier-based business model" in which the electronic module suppliers participate in the product development.

What are the advantages in using the "supplier-based business model"?
- Allows parallel development which reduces overall vehicle development time.
- Allows scaleable vehicle content or features to be added as options.
- Moves a portion of the complexity outside the OEM into the supplier base.
- Allows sharing of product behavior models to further reduce development time.

Modeling must support an increase in requirements

Two Main Components of a Distributed Product Architecture
- **Application** – the customer perceived product behavior that is distributed across the members of the automotive electronics system.
- **Network** – the communication scheme that glues the application portions together.

Requirements - when centralized

| Application |

Requirements - when distributed

| Application | Network |

Observation: Modeling has its original roots in centralized product architectures - what modeling tools embrace the distributed product approach?
Key ECU automotive software elements

- **Operating System (OS)** – the task scheduling (event, periodic, etc.)
- **Application (APP)** – supports normal powertrain operation, diagnostics, calibration, and manufacturing modes
- **Network (NET)** – communication data transfer, OEM network strategy (FNOS, GMLAN, etc.), and file transfer

Use of embedded operating systems is increasing

- Both the OSEK Operating System and other larger RTOS solutions are available

Advantages of an operating system:
- Portability
- Scalability
- Reusability
- Application-independent
- Network-independent
- Configurable
- Provide optimal adjustment of the architecture
- Reduces supporting infrastructure in contrast to making your own OS

- The U.S. popularity of “off-the-shelf” Operating Systems is rather low with exception of telematic-specific ECUs
The influence of non-competitive technology

- Competitive and non-competitive – what is the difference?
- The boundary between competitive and non-competitive is important for automotive - but is very different in the mind of every engineer
- Any "agreed upon non-competitive technology" usually gains standardization
- Today – the boundary has a strong influence by both the engineering community and by management (a management that is not well connected to the technology side of software)

- For software engineering, what does competitive/non-competitive mean?
- Isn’t a switch debounce algorithm essentially non-competitive?

- Today the software elements – the Network and the Operating System are considered as non-competitive

Major distributed product modeling requirement influences

Major Influences:
- Distributed functions
- Small area networking – perhaps multiple networks
- Communication protocols – more than CAN
- OEM-specific network operating strategy
- OEM-mandated embedded software components

- Model-based development requires a strong systems engineering focus
Distributed functions influence modeling

To distribute a basic function: PARTITION

- Input process
- Control process
- Output process

split the function between different processes

IN  I  C  O  OUT

IN  I  Tx  Rx  C  O  OUT

Distributing a function adds:
- 2 new processes, transmit and receive AND a communication system

With most modeling concentrating on just the C-block, how do you also model the network activities?

Each element of a distributed function influences the model

- Implementation requires software, hardware, information engineering, and a communication transfer over the selected medium (typically wire)

What portions do we need to model?

- Because the timing of the distributed function is primarily controlled by software and the software scheduling (operating system) –

How do we model ECU-specific software timing behavior?
Modeling a distributed function – Car Door Unlocking

Basic Function - Door Unlocking

Distributed Function - Door Unlocking

How do you model across different nodes or modules?

Distributed function conversation influence modeling

- The "transfer dialog" which interconnects the information transfer portion of the distributed function must become a portion of the model

- **Event**
  - transfer information once

- **Repetitive, Periodic or Cyclic**
  - transfer information continuously

- **Conditionally Repetitive**
  - transfer information repetitively while condition is true

How do you model the conversational behavior of the network?
The use of multiple networks impacts modeling

- 3 to 5 networks are common across many car companies
- System partitioning splits motion control (engine area) from "body" electrical functions (driver & passenger area)
- Typically – the air bag subsystem is separated
- Entertainment subsystem is connected to existing body bus or may be separated (for example using MOST)
- This is an example of one Ford vehicle architecture that uses 3 CAN networks

How do you model across multiple networks?

Model-based development covers all portions of the function

- Whether distributed or not
- Whether one or more bus connections are used
Choice of communication protocol impacts modeling

- Automotive communication choices include – CAN, LIN, FlexRay, MOST, etc...
- Automotive CAN physical layers come in 4 flavors
  - DC, Ford, and GM High Speed - 500K bps without shield (1050 type transceiver)
  - nearly identical to J1939 physical layer
  - DC Fault Tolerant – 83.3K bps (1054 type transceiver)
  - Ford Medium Speed – 125K bps with WakeUp/Sleep (1040 type transceiver)
  - GM Single Wire – 33K bps with WakeUp/Sleep (5790 type transceiver)
- LIN is a UART-based master/slave protocol
- To handle LIN specification ambiguities, Ford and GM use the recently developed SAE J2602 instead of LIN 2.0
- For the auto industry - Each serial data communication protocol and its selected physical layer influences the use of modeling

Observation: Although high speed CAN is extremely stable, the auto industry seems to continue down the path of physical layer and protocol selection churning.

Changing communication protocols influence modeling

Today's communication protocols include -
- J1850 – Chrysler, J1850 – Ford, J1850 – GM
- CAN – DC Net (DaimlerChrysler), CAN – FNOS (Ford), CAN – GMLAN (GM)
- LIN 1.2, LIN 2.0

Recent communication protocol additions -
- J2602 – an upgraded LIN variation (supported by GM & Ford)
- SENT – a new PWM protocol (originated by GM - throttle control application)

Current competing protocol candidates for X-by-wire -
- FlexRay - the current leading contender
- CAN – recently FAA-approved for flight control systems
- Time Trigger Protocol (TTP)
- Time Triggered CAN (TT-CAN)

How do you model with each different protocol choice?
**CAN**

- Very stable protocol - over 15 years old
- Wide availability of silicon
- Excellent undetected error rate
- The ability to use system engineering principles to correct any CAN deficiencies
  - Both speed and distance limitations can be addressed

**Probability of an undetected CAN standard frame**

\[ p < 4.7 \times 10^{-11} \times \text{error rate} \]

For example -

If the error rate is 1 bit error every 700 milliseconds at a bit rate of 500 Kbps over an interval of 8 hours per day (365 days per year) - then the statistical average is 1 undetected error in 1000 years

Undetected error rate source – CiA: CAN in Automation

---

**LIN – Local Interconnect Network**

- Based on the common UART (Universal Asynchronous Receiver Transmitter)
- Provides slow transfer rates between 10K-20K bits per second
- Uses a master-slave control strategy
  - Master asks and then Slave responds
- Slaves do not require a crystal
- Uses a single wire physical layer with 0 to 12 V signal transitions
- Uses a software-based protocol (if implemented in a microcontroller)

LIN is being considered for some powertrain applications -
- Simple fan/cooling controls as an engine control sub-bus
- Slow speed sensors

- SAE J2602 is a Ford and GM-specific derivative of LIN – that focuses on ASIC-based modules rather than software-based
**FlexRay**

- FlexRay – the leading contender for automotive x-by-wire applications
  - Essentially a TDM (Time Division Multiplex) protocol
  - Can be used for time-triggered and/or event-triggered communications
  - Up to 10 Mbps
  - Currently supports redundant communications with a "bus guardian" feature
  - Heavy emphasis on determinism rather than event-based probabilities
  - Major contributors include BMW, DC, GM, and Freescale
- BMW already has used this technology - suspension control subsystem
- Future automotive subsystems like stability control and new propulsion systems may require higher speed data transfers
- Is a new version of FlexRay coming?
  - Is the "bus guardian" feature being removed?
  - Is support for redundancy being removed?
  - Just a high speed replacement for CAN?

---

**Influence of OEM-specific Network Operating Strategies**

The Network Operating Strategy (NOS) defines the requirements for the network portion of the automotive distributed embedded system including:

- What communication protocol is used and its electrical interconnect
- How to define network data, network messages, and network addresses
- How and when to transfer network data
- How to initialize the network and optionally "go to sleep" and "wakeup"
- How to transfer large files or provide flash programming
- How to determine and handle loss of network or loss of relationship
- The OSI 7-Layer Model is one structural organization that many OEMs use to describe major portions of the their Network Operating Strategy requirements
OEM NOS Implementation = Network Software Components

- **Interaction Layer** – provides a signal (just the data) interface rather than a message interface
- **Diagnostics** – handles the diagnostic services related to communications
- **Transport Protocol** – handles the transfer of files – especially diagnostic information and flash programming
- **Network Management** – determines which other nodes are the bus
- **CAN Driver** – directly controls the transmission and reception of CAN message using the CAN protocol controller

Network Operating Strategy Requirements & Implementation
**Network operating strategy impacts modeling**

- The Network Operating Strategy (NOS) defines the requirements for the network portion of the product’s distributed embedded system.
- The Network Operating Strategy which specifies a portion of the system behavior can also be described using models.
- Once developed, models of network operating strategy can be re-used for the entire distributed product development community including the individual module developers.

**The NOS implementation impacts the modeling**

**KEY CONSIDERATIONS**

- If the NOS software implementation is identical for each module, then a "single" model of the NOS software behavior can be shared across the enterprise.
- If the NOS software implementation executes the network behavior every 10 milliseconds (for example), then a model that includes this software scheduling behavior can be shared across the enterprise.
- If the NOS implementation uses a common vehicle-specific messaging database, then a model that includes access to this database can be shared across the enterprise.
Number of OEM-specific Network Operating Strategies

- Each company has their own NOS solution
- Each solution is nearly identical in concept
- Supporting each OEM NOS is costly at the industry level – especially for each ECU module supplier

Industry use of OEM-specific common software solutions

- Early J1850 to today’s CAN common software

WHAT’S NEXT?
AUTOSAR – AUTomotive Open System ARchitecture

- AUTOSAR is an industry collaboration
- Open standard for automotive E/E architectures across all vehicle domains
- A basic software infrastructure for the management of automotive applications – all current and future applications

Key Elements of AUTOSAR

- One industry-level group to develop, manage, and maintain the effort
- One common set of requirements
- A collection of commercial available implementations
- A common process of implementation – to foster the creation of common tools

- As a potential industry-wide software solution AUTOSAR is essentially an agreed upon non-competitive technology

AUTOSAR – Business Case for OEMs

- Exchangeability – across OEMs and suppliers

![Diagram showing exchangeability between different OEMs and suppliers]
AUTOSAR – Business and Technical Goals

- Reduce development time
- Reduce time to market
- Reduce resources
- Decrease level of complexity
- Satisfy future innovative vehicle applications and contemporary automotive E/E functions
- Encourage competition on innovative functions by reducing the emphasis of non-competitive technologies like vehicle network communications and operating systems
- Minimize the current business barriers between functional domains. (Provide the ability to map functions and functional networks to different control nodes in the system, almost independently from the associated hardware.)
- Increased use of commercial off-the-shelf software and hardware
- Provide solutions to conflicting requirements from governmental entities - key items include environmental aspects and safety requirements

AUTOSAR – Business and Technical Goals (continued)

- Provide standardization of basic system functions as an industry wide "Standard Core" solution
- Provide scalability to different vehicle and platform variants
- Provide easier integration of functional modules from multiple suppliers
- Allow the transferability of functions across the distributed product domain
- Handle availability, safety, and redundancy requirements for x-by-wire and other safety-critical subsystems
- Provide ability to accomplish software updates and upgrades over vehicle lifetime
- Provide maintainability throughout the whole "Product Life Cycle"
AUTOSAR – Business Cooperation

**AUTOSAR partnership is an alliance of OEM manufacturers and Tier 1 automotive suppliers**

- August, 2002 - BMW, Bosch, Continental, DaimlerChrysler and Volkswagen conduct initial discussions - soon afterwards Siemens VDO
- November, 2002 - establish joint technical implementation strategy team
- July, 2003 - Core Partners signed off
- November, 2003 - Ford Motor Company joins
- December, 2003 - Peugeot Citroën Automobiles S.A. and Toyota Motor Corporation
- November, 2004 - General Motors joins
- The AUTOSAR project plan foresees the completion of its initial test and verification phase in August 2006.

AUTOSAR – Working groups

- Approximately 50 working groups (WG) are established to handle AUTOSAR responsibilities and develop requirements documents including –
- Standardized network operating strategy and gateway interfaces across the different vehicle communication protocols
- Standardized operating system interfaces
- Standardized subsystem and ECU interfaces across the different vehicle domains
- Definition of requirements and analysis of existing solutions in the area of basic software modules and automotive operating systems
- Definition of data exchange formats for describing necessary elements of a vehicle’s E/E system architecture

- Each working group is staffed by the AUTOSAR member companies who represent the extensive knowledge and experience of the partnership members from the various automotive domains.
- The entire effort includes approximately 250 full-time expert members
AUTOSAR Requirements

- Much bigger than today's Network Operating Strategy approach

AUTOSAR – Includes Standardized Interfaces

- Standardization of different APIs to separate the AUTOSAR software layers
- Facilitate encapsulation of functional software components
- Definition of the data types of the software components to be AUTOSAR compliant
- Identify basic software modules of the software infrastructure and standardize their interfaces
AUTOSAR – RunTime Environment

- AUTOSAR’s RunTime Environment or RTE embraces the standard use of an Operating System
- AUTOSAR runtime environment to provide inter- and intra-ECU communication across all nodes of a vehicle network
- All entities connected to the AUTOSAR RTE must comply with the AUTOSAR specification
- Enables the easy integration of customer specific functional SW-modules

AUTOSAR – Provides Choices

- Provides standard interfaces to support a variety of choices
AUTOSAR – Implementation Software Components

- Software component implementations will include networking, operating system, and general purpose application functions
- Target is to have software components available from a variety of software suppliers

AUTOSAR Software Components

AUTOSAR Software Components = Application Generic Software Components + Operating System Software Components + Network Software Components

AUTOSAR – Software Development

- Starts with a Library of AUTOSAR components
- Use the “generation tool” to select what features you want – including vehicle messages and perhaps powertrain application information
- Add the “generated” AUTOSAR code to the application code
- Compile, test, etc... until complete
- Example shown is for powertrain application
AUTOSAR – Supports Variety of Communication Methods

AUTOSAR includes...

- Supports single communication channel
  - CAN

- Supports multiple bus configurations with gateway
  - CAN
  - LIN/2002

- Supports redundant bus configurations
  - FlexRay1
  - FlexRay2

AUTOSAR – The Application Specific Building Block

- Allows rapid development
- Allows flexibility of functional partitioning
- Provides a larger palette of design options for powertrain applications
AUTOSAR – Increases Software Standardization

AUTOSAR standardizes only the non-application infrastructure side of automotive embedded software.

- AUTOSAR increases the value of non-competitive software
  - Automotive embedded software must handle both the application side (feature content) and the non-application infrastructure side (networking, operating system, etc.).
  - Software complexity can be reduced by standardization – especially if concentrated on the non-application infrastructure side.
  - **Standardizing a software architecture – one that encompasses:**
    - common vehicle network solutions
    - common use of operating systems
    - common use of generic I/O
    - can decrease development time and increase the value of model-based development
  - AUTOSAR – is an excellent business and technical direction to accomplish this standardization.
AUTOSAR will reduce development time

- AUTOSAR embraces the non-competitive system domain
- From a modeling-centric view, AUTOSAR targets COMPATIBILITY and CONSISTENCY across the modeling development

Simulink Styleguide for AUTOSAR

- AUTOSAR WP1.2 working committee
- Current document is in draft form
- Provides a styleguide for modeling AUTOSAR software component features in Simulink
- Targets how AUTOSAR software component fundamental metaclasses can be expressed in Simulink
- Covers the subset definition as defined in the "Interaction of Behaviour Modeling" from a simulation and code generation standpoint
- Next steps

WP1.2 working committee members
- Christian Dziobek (DaimlerChrysler AG)
- Dr. Thomas Ringler (DaimlerChrysler AG)
- Dr. Richard Thompson (The MathWorks)
- Dr. Uwe Honekamp (Vector Informatik)
Model-based development desired outcomes

- Models that demonstrate the requirements or behavior of the vehicle
- Models that demonstrate the requirements or behavior of the module
- Models that deliver auto-code generation
- Models that support testing of the vehicle
- Models that support testing of a vehicle subsystem
- Models that support testing of the module – that support incremental integration
- Models that demonstrate the requirements or behavior of the communication system (protocol + physical layer + data transfer) – whether single or multi-bus configuration
- Models that demonstrate the behavior of the customer (driver and passengers)
- Models that demonstrate the behavior of the driving environment (the plant model)
- Models that support the servicing of the vehicle/module

**AUTOSAR will likely impact most of these modeling domains**

Automotive distributed embedded systems engineering

- Who supports the OEM-specific AUTOSAR selections for the distributed embedded system
- Who supports the modeling of the various vehicle communication protocols
- Who supports the information transfer across multiple bus – via gateways
- Who supports the various software scheduling methods used across the variety of modules
- Who will manage the technical and business side of enterprise-wide modeling

- Improving the systems engineering approach is required across the industry
- Good systems engineering will increase the value of model-based development
Automotive industry needs from the academic community

Distributed Embedded Systems Engineering (DESE) should become a new academic discipline - that combines...
- Computer science
- Electrical engineering
- Distributed systems engineering
- Information engineering

- Discuss the competitive/non-competitive business boundary
- Include "the business side" as a portion of the traditional engineering problem domain
- Construct a "common" set of generic terminology that can be used across the academic community as well as the automotive industry

*Such actions will improve the auto industry adoption rate of model-based development*

Contacts

- Presenter
  - Bruce Emaus: bruce.emaus@vector-cantech.com