

# Computer-aided Architecture Design & Optimized Implementation of Distributed Automotive EE Systems

Antal Rajnak  
Mentor Graphics  
Prinz Eugen Str. 72  
Vienna, Austria  
+41-79-373-8601

antal\_rajnak@mentor.com

Ajay Kumar  
Mentor Graphics  
8005 SW Boeckman Road,  
Wilsonville, Oregon, USA  
+1-503-685-1055

ajay\_kumar@mentor.com

## ABSTRACT

Complexity of Automotive EE systems has grown exponentially during the past decade. Model-based function design and simulation are widely accepted and used by the Automotive EE community of today, to face some of the challenges. Emerging new standards - like AUTOSAR - bring standardized interfaces to low-level software, reducing cost and most importantly introducing a new level of abstraction to function implementers. Ease of software component reuse is another novel benefit brought by AUTOSAR. However, there is little guidance on how to translate the results of model-based function design into robust and efficient system implementations in a highly distributed environment. This paper outlines a proposed system level design methodology, supported by a set of point-tools, to bridge the Implementation Abyss. The flow covers: model transformation, architecture design, scheduling of networks and tasks distributed across multiple ECUs, harness design - followed by metrics and criteria-based evaluation of the resulting architecture. Automated creation of AUTOSAR-compliant configuration files for ECU generation is provided to complete the process.

## Categories and Subject Descriptors

C.0 [General]: System architectures  
C.2.1 [Network Architecture and Design]: Network topology  
D.2.2 [Design Tools and Techniques], Computer-aided software engineering

## General Terms

Design

## Keywords

AUTOSAR, System-Level Design, Scheduling, CAN, LIN

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2007, June 4–8, 2007, San Diego, California, USA.

Copyright 2007 ACM 978-1-59593-627-1/07/0006 ...\$5.00.

## 1. INTRODUCTION

Automotive Electrical/Electronic systems are rapidly increasing in size, communication complexity and software content. Today's vehicles can have more than 60 ECUs, connected by multiple communication buses and running millions of lines of software. Further, much of a car's most sophisticated functionality – electronic stability control, active cruise control etc – requires that multiple ECUs interact. Existing single-ECU-focused design methodologies no longer suffice; system-level design is now required.

The emerging AUTOSAR standard (with some suggested enhancements) provides a technically and economically viable target for a system-level automotive E/E design methodology. Technically, the extensive AUTOSAR metamodel and its rich set of interface definitions allow system-level E/E architectures to be expressed in non-proprietary formats. Economically, AUTOSAR opens up a large, unified market, making creating design tools viable.

This paper describes a system-level design methodology targeting AUTOSAR based on point-tools. We begin by introducing AUTOSAR. We then provide an overview of the suggested tool set and describe its “correctness by design” capability. Where applicable, we describe benefits of links to physical wiring and function modeling tools. We detail an already-deployed point tool in our flow, the Network Cluster Builder. Finally, we suggest enhancements to the AUTOSAR standard to improve its applicability to system-level design.

## 2. BASIC AUTOSAR CONCEPTS

The AUTOSAR (AUTomotive Open System Architecture) initiative is a partnership of automotive manufacturers, suppliers and tool-vendors aiming to standardize an open software architecture for automotive electronic control units (ECUs). AUTOSAR specifies a layered software architecture that clearly defines interfaces between application software components (written to realize user-visible vehicle functionality) and infrastructure components. Infrastructure components are then tightly specified to allow implementations to be developed by

different vendors. Interfaces with controllers and peripheral hardware are also defined.

User-visible vehicle functionality is implemented through interconnected application Software Components (SWCs). Atomic SWCs run entirely on one ECU; composed SWCs are hierarchical compositions of atomic SWCs. SWCs interact with their environment only through ports. The abstraction of interconnected SWCs of the entire vehicle is called the Virtual Functional Bus (VFB). Ports are used for communication within an ECU, between ECUs as well as with basic software (BSW) services that have a port. Thus, the application SWC is isolated from dependencies on particular hardware, allowing it to be relocated to different ECUs.

The VFB is implemented on a particular ECU through the Run Time Environment (RTE). For efficiency, the RTE is created for the exact combination of SWCs running on a particular ECU.

An AUTOSAR system [3] is described through various XML files: SWC Descriptions, ECU Resource Descriptions, System Configuration Description etc. These files describe various attributes of SWCs and how they are mapped to ECUs. While the syntax and semantics of these files are defined by AUTOSAR, the methodologies through which they are created are left to tool vendors.

### 3. PROPOSED TOOL FLOW

We describe our overall tool-flow as a Vehicle System Architect flow that feeds into a Vehicle System Builder flow.

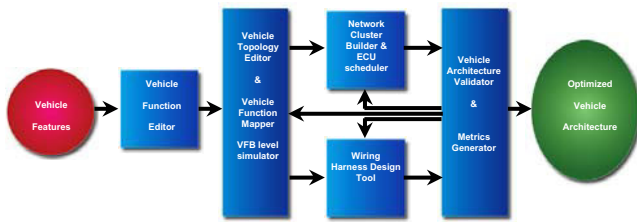


Fig. 1: Vehicle System Architect (VSA) Flow

#### 3.1 Vehicle Function Editor (VFE)

The VFE is the entry point into our proposed flow. A vehicle’s electrical system is decomposed by most OEMs into 100-200 vehicle functions. Users create atomic or composed SWCs that represent their vehicle functions. This is done using the VFE either through manual graphical input or through import from function modeling tools like Matlab/Simulink.

Note that it is not essential to have a complete functional description at this stage; even just the “envelope” of the component describing the interfaces it provides and requires is useful.

Specification and development of SWCs is highly distributed in time and space. Since SWCs enter our flow from many sources, consistency checks are necessary to catch errors early. Even with

just interface descriptions, static checks for inter-component interface consistency can be performed. Additionally, end-to-end timing requirements (necessary for some downstream tools in our flow) can be entered in the VFE. The VFE outputs AUTOSAR SWC Description files.

#### 3.2 Vehicle Topology Editor (VTE)

Parallel to the vehicle function decomposition described above, the physical architecture of the distributed network in the car is defined in the VTE. ECUs, sensors and actuators connected to them and communication clusters connecting the ECUs are instantiated.

Since almost every car project reuses ECUs from previous projects, some provision must be made here to import descriptions of ECUs from a company-wide library. Also, at this stage, a user may want to instantiate “soft ECUs”, in which key attributes and hardware interfaces are specified (e.g. CPU type, communication interfaces, I/O interfaces) but are not fixed pending function mapping described below. This enables more degrees of freedom during mapping.

This point in the flow should also support links to wiring harness design. Sensors and actuators are usually in well-defined places in a vehicle, determined by mechanical and environmental constraints and car company practice. Since sensor and actuator connections to ECUs are specified, estimates of wiring requirements can begin to be made. Also, existing harness designs can be imported into the VTE in case a car project is largely based on a preexisting design.

Note that at this point in the flow, while physical interconnection possibilities have been defined, we don’t yet know what the actual communication requirements are, since we have not yet mapped SWCs to ECUs.

#### 3.3 Vehicle Function Mapper (VFM)

The VFM is used to map a set of vehicle functions (represented by a collection of SWCs) to the logical ECU and gateway network defined in the VTE. Given available resources on the ECU (described in the ECU Resource Template) and required resources for SWCs (described in the Software Component Template), the VFM can determine whether a proposed mapping is legal. “Soft ECUs” instantiated earlier are fixed by choosing from a library of available real ECUs.

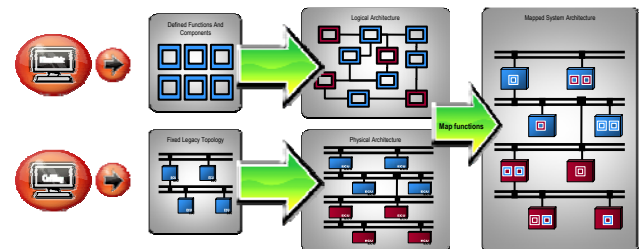


Fig. 2: Topology Editing and Function Mapping

Where vehicle functions are distributed over several ECUs, communication gateways can be identified. Since end-to-end communication needs can be identified, timing requirements can now be split among SWCs, buses and gateways. Fully automatic gateway configuration can be performed. As well, system-level application signals can be identified and assigned to cluster-level communication IDs.

The VFM generates an AUTOSAR-specified System Configuration Description file that describes the system’s topology, mapping and communication requirements. Since end-to-end connectivity needs are now known, The VFM can also drive wiring harness tools, especially tools that automatically generate wiring based on connectivity requirements and harness restrictions. Also, the VFM could generate an ECU-specific view (called an ECU extract) of the system description. This is used in configuring and building the software image for a particular ECU.

### 3.4 Virtual Function Bus-Level Simulator

The VFB-level view of the vehicle E/E architecture is a network of interconnected SWCs communicating with each other and with their environment through ports. If PC-compiled versions of the SWCs are made available, this network can be simulated to verify that a particular set of atomic SWCs does indeed perform the intended vehicle function. This can be used to verify both the software image on a single ECU as well as vehicle functions distributed across multiple ECUs.

### 3.5 Network Cluster Builder (NCB)

Here users schedule communications on CAN, LIN and/or FlexRay clusters (buses) isolated to that particular cluster, defined during topology entry in VTE. (End-to-end communication requirements were split over the buses used during the function mapping process.)

The NCB takes partial cluster definition files (describing at least the ECUs connected to and the system signals transferred within the cluster). A complete cluster definition file is then created either manually, semi-automatically using analysis or fully automatically using synthesis. The NCB can account for existing complete communication module configurations of fixed ECUs.

More details on the underlying theory and use modes of the NCB are presented later in this paper.

### 3.6 ECU Scheduler

The *AUTOSAR OS* definition is based on the OSEK RTOS, using pre-emptive scheduling of tasks. (The lowest conformance level requires OSEK RTOS, extended with time and memory protection for the higher conformance levels.)

There are several prerequisites to perform task schedulability analysis. All software functions must be grouped into tasks (or interrupt service routines), with well defined maximum activation rates and priorities. Worst-case execution time (WCET) values of each task must be known. These parameters are defined in the

ECU configuration description file, an input to this tool. However deadline requirements (the time from task activation to the end of task execution) are missing from the current AUTOSAR timing requirements specification. Transformation from end to end timing requirements to deadline requirements for tasks and network frames shall be described by a unified timing model, which is under development.

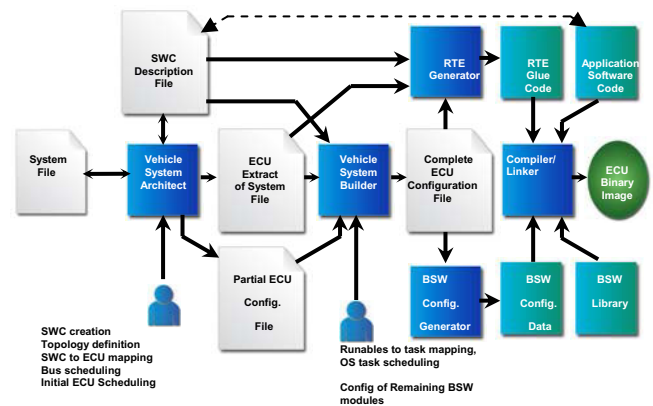
Combining network and task scheduling capability within the same tool enables predictable scheduling of distributed tasks running on multiple ECUs in a multi-protocol network environment.

### 3.7 Metrics Generator – Cost Analysis (MG)

The Metrics Generator (MG) compares and ranks alternative solutions for a given set of requirements, ideally in terms of resulting life-cycle cost for a given product line. A product line is characterized by number of cars produced, quantity and feature content of variants, and the distribution of options. Factors considered beyond direct component costs are scalability, flexibility and extensibility of the architecture during its lifetime, if criteria and cost models [4] for these are available. Using the MG, comparative (as opposed to absolute) monetary cost analysis can be performed in the early phases of a design, when no detailed data are available. A set of alternative architectures for a product line can be compared in terms of life cycle cost. At this point, detailed harness cost can be obtained from the harness design environment, and considered in the evaluation. Sensitivity analysis based on a series of scenarios with respect to volume change or shift in variant volumes can be performed. Reports with detailed score data can be generated.

A simplified analysis could be run constantly in the background, measuring the impact of any action taken by the operator during creation of the architecture, providing “live” feedback to the user. This information is presented directly in the Function Mapper and the Network Cluster Designer tools.

### 3.8 Vehicle System Builder (VSB)



**Fig. 3: Vehicle System Builder Flow**

The Vehicle System Builder is an umbrella for the point-tools necessary to build the complete software for AUTOSAR ECUs.

### 3.8.1 BSW Configuration Editor

AUTOSAR has defined three different types of XML files to describe the configuration of the Complete Basic Software (BSW). The complete BSW includes 30-40 BSW modules configurable independently from each other. However, to cope with scenarios where some functions might be implemented in more than one module, special consistency checks must ensure that these modules are configured in a consistent manner. The following three files must be handled by the BSW configuration editor:

- ECU parameter description (EPD) file for BSW configuration
- BSW module description (BMD) file includes the vendor specific parameters defined by the BSW module supplier, the actual values for all pre-compile time configurable AUTOSAR, and vendor specific parameters. This file can be seen as the contract between supplier, and car OEM for building the BSW module library when object code is being delivered.
- ECU parameter configuration (EPC) file describing actual values of the parameters contained in the EPD and BMD files

A serious headache for AUTOSAR users is the fact that different sections of the EPC file might be created by different actors in the process. E.g. configuration data for all COM modules are created by a car OEM using the network builder tool, configuration data for the other modules being hand edited by different Tier I-s with a BSW configuration editor. To manage this scenario a generic merger function must be part of the editor functionality, to support easy concatenation of configuration data coming from the different sources.

### 3.8.2 Configuration Generators for BSW modules

The definition of the AUTOSAR BSW layer contains nearly 40 independently configurable modules. A dedicated configuration generator is defined for each independently configurable BSW module that takes the complete EPC file as input, extracts BSW module specific information and generates all necessary configuration data

### 3.8.3 Locator for “Post-Build reloadable” configurations

.At this stage, special provisions could be made to support changing configuration data without requiring a rebuild of the entire ECU image. Configuration generators for BSW modules do generate \*.h and \*.c files, holding configuration data. In case of Post-build re-loadable configurations this would not be sufficient, because updated configuration data would still need to be compiled together with the entire ECU SW. Any post-build configuration must have a fixed and agreed location (usually a separate flash sector which can be erased and re-loaded independently from the rest of the software). The locator tool groups post-build re-loadable configuration data and creates a single output file in Motorola-S or Intel Hex format. This file can be downloaded into the ECU without recompilation of the entire application SW.

### 3.8.4 Run-Time Environment Generator

The RTE generator generates the glue code connecting application software (a set of SWCs) to the required (sub)set of BSW. The tool is to be used in two different phases of vehicle SW development:

- *Contract phase*: the SWC API is generated from the SWC description file.
- *Generation phase*: glue code is generated connecting the application SWCs and BSW, using the EPC file as input.

The generated glue code resolves port-oriented API calls in SWCs into:

- COM messages (system signals or signal groups)
- Global variables for data exchanged
- Function calls

Significant optimization is obtainable if SWCs are available in source form.

## 3.9 Network Cluster Builder Details

For well over a decade, design of multiplexed networks in automobiles has with very few exceptions been a bottom-up style engineering exercise. This approach, focusing on agreed “messages” between the ECU-s directly involved, has left the effects of unintentional interactions on the bus to be resolved by testing. “Build-and-test” has been the prevailing paradigm. Faults found in late phases of the car project were expensive and risky to correct, associated with high risk for introducing secondary errors. Another shortcoming of this method is lack of flexibility, preventing reuse of previously allocated bandwidth.

## 3.10 Scheduling Techniques for Networks

Since the mid-1980s there has been a branch of computer science research into mathematics that determines the longest time it can take a task to run. The most successful family of analysis is called *response time analysis* and applies to systems scheduled by fixed priorities. It has been used to analyze traffic on a CAN bus, for CPU scheduling etc.

Controller Area Network (CAN) buses arbitrate between messages by using priorities; each message has a unique priority (that is also the message’s identifier). A frame inherits the timing requirements of the signals packed within it.

Deadline Monotonic Analysis (DMA)[4] is a deterministic scheduling technique that can (under certain assumptions – bounded frame set, bounded frame size, fixed identifiers, minimum frame periodicity, priority-based CAN controllers[6]) find worst-case message latency (WCML) on priority-based buses like CAN. WCML is defined as the longest time between placing a frame in the CAN controller and receiving it at all nodes. The analysis takes retransmission of corrupted frames into account. Solutions for scheduling LIN [9]and FlexRay using similar techniques have also been deployed.

Using protocol-specific deterministic scheduling techniques, reliability and resource utilization (i.e. bandwidth, interrupt load, RAM usage) of a networked system can be optimized.

### 3.11 Towards Correctness-by-Design

The Network Cluster Builder tool can be used at three distinct operational levels:

- Manual operation
- Analysis - of manually-created configurations
- Synthesis - fully automatic requirements-based generation of the complete network configuration

In manual mode, users create/modify PDU-s (frames) and related parameters (frame ID, transfer mode, and the timing parameters). Users interactively pack system signal/signal groups to Protocol Data Units (PDUs) and create or modify communication schedule tables. Strict consistency checking for all data objects used by the cluster is performed on the fly.

The concept of timing analysis is not something that is well accepted by the automotive industry. Instead of using this term we should better refer to *simulation*, when the tool is actually performing what we would normally call analysis. In analysis/simulation mode, the NCB tool is run on an interactively-created communication matrix to calculate worst case message latencies for PDUs sent over any of the buses. Deadline requirements are derived from the vehicle function end-to-end timing requirements.

Finally, in fully-automatic mode, the NCB tool can perform complete cluster synthesis based on requirements declared in its input files. This mode offers fully automatic PDU (frame) creation, signal packing and schedule table generation. The tool accepts a set of control parameters (e.g. available bandwidth, RAM usage and interrupt load) - with relative weights attached on a per ECU basis, to control the algorithm.

Task scheduling for individual ECUs using the AUTOSAR RTOS - supporting preemptive priority based scheduling of tasks - can be managed with similar techniques available today.

The synergies achievable by combining the two scheduling processes are significant, enabling the user to engineer predictable distributed systems in a multi-protocol network environment.

### 3.12 Suggested Improvements to AUTOSAR

The AUTOSAR Timing Meta Model document (02.02.2005, version 0.91) is an early attempt to provide a timing model for systems built on the standard. The document attempts to deal with both *timing requirements* as well as *timing behavior* of AUTOSAR objects.

The current AUTOSAR timing model document fails to address some important concepts like jitter, phasing and precedence requirements. Deadline requirements (the time from task activation to the end of task execution) are not defined.

These are essential for successfully dealing with complex systems like an automotive E/E system described using AUTOSAR. Timing behavior information is spread over several abstraction levels and across several organizations (function developers,

software component developers, ECU developers, system integrators, basic software component developers and even silicon vendors).

Bringing this distributed timing behavior information together in a model that makes accurate predictions will, using the present definition of the standard, be difficult. Continued effort in further developing the existing timing model, as well as bringing cooperative design processes into play, will, in a future release of the AUTOSAR standard, improve this situation.

Until then, scheduling analysis can and should be applied to areas of the standard with well-understood requirements and behavior, such as task scheduling under the AUTOSAR OS and scheduling of communication networks.[2]

A consequent use of automated tools implementing deterministic scheduling will bring significant improvements in reliability and efficiency and lead to repeatable processes.

### 3.13 Conclusion

Rapid and parallel development based more on **virtual development** and less on physical prototypes is becoming more important in automotive development. The proposed methodology supports **virtual development** and **design to correctness** through the tool-chain outlined – parts of which has been already deployed and successfully used in production [1], [8].

AUTOSAR, by representing a global standard opens up a sizeable market for tool developers, significantly bigger than the one represented in the past by individual car OEMs, each with proprietary solutions and needs. The Electronic Design Automation (EDA) industry is turning its attention to Automotive. Many of the problems facing the automotive system designer today are not different from those challenges solved in the past in other industry segments, like semiconductor and telecom. Technologies successful in addressing these previous challenges can now be applied to Automotive E/E design in the context of the flow outlined in this paper.

### 3.14 ACKNOWLEDGMENTS

We thank Serge Leef and Istvan Horvath for some of the figures in this paper and for their valuable feedback on its content.

### 3.15 REFERENCES

- [1] Antal Rajnak and Mats Ramnefors “*The Volcano Communication Concept*” SAE Conference 2002, paper 2002-21-0056 <http://www.sae.org/technical/papers/2002-21-0056>
- [2] Antal Rajnak. “*A Structured Engineering Approach to Automotive Network Design*”, Euroforum Symposium on Systems Engineering, Munich, 2006
- [3] AUTOSAR *documents and specifications* <http://www.autosar.org/>
- [4] A. Ghosal, S. Kanajan, R. Urbance and A. Sangiovanni-Vincentelli, “*An Initial Study on Monetary Cost Evaluation for the Design of Automotive Electrical Architectures*”, SAE

Conference 2007.

<http://www.sae.org/technical/papers/2007-01-1273>

- [5] K. Tindell and A. Burns, “*Guaranteeing Message Latencies on Controller Area Network (CAN)*”, Proceedings 1st International CAN Conference, 1994, pp 2-11.
- [6] K. Tindell, H. Hansson and A. J. Wellings, “*Analysing Real-Time Communications: Controller Area Network (CAN)*”, Proceedings 15th IEEE Real-Time Systems Symposium, San Juan, Puerto Rico, 1994, pp 259-265.
- [7] K. Tindell, A. Rajnak and L. Casparsson, “*CAN Communications Concept with Guaranteed Message Latencies*”, SAE Paper 98C050, 1998.
- [8] L. Casparsson, K. Tindell, A. Rajnak and P.Malmberg, “*Volcano – a revolution in on-board communication*”, Volvo Technology Report, 1998.  
<http://www.volvo.se/rt/trmag/vtr981/article2/a2no198.html>
- [9] W. Specks, A. Rajnák, “*The Scaleable Network Architecture of the Volvo S80*”, 8th Intl. Conference on Electronic Systems for Vehicles, Baden-Baden, Oct 1998, pp. 597-641