Specification and Design of Electronic Control Units

Jürgen Bortolazzi, Thomas Hirth, Thomas Raith
Daimler-Benz AG F1M/E
D-70546 Stuttgart
e-mail: bortolazzi@str.daimlerbenz.com

Abstract

Electronic control units (ECU) play a more and more important role in the development of road vehicles. Forecasts lead up to 25% ofthe total vehicle value in 2000. The increasing complexity and stringent quality and cost requirements mandate tremendous improvements in the specification and design process. This paper presents the cooperative activities at Daimler-Benz research and Mercedes-Benz development departments to install an optimized design process.

1 Introduction

In order to improve safety, to reduce emmission and fuel consumption or to improve comfort and driver information, a rapidly increasing number of vehicle components are controlled by open-loop and closed-loop systems, which consist of sensors, actuators and electronic control units (ECU). Typical examples are powertrain control systems such as engine and transmission management, active suspension and braking systems such as ABS (Anti Blocking System) and ABC (Active Body Control) or navigation systems. The development process of such systems is illustrated in figure 1, where three major levels can be identified:

1. the mechatronic vehicle system, such as an engine management consisting of the engine, sensors and actuators as well as the ECU
2. the ECU typically consisting of a standard microcontroller (e.g. Siemens 80C167), specific signal processing and controller IC's as well as the necessary packaging, power supply and power electronics components
3. the control software, consisting of open-loop and closed-loop control algorithms, communication and management functions as well as onboard diagnostics.

The requirements on the performance of ECUs have increased dramatically over the last ten years: From isolated control units simply controlling specific vehicle components in the 80's to autonomous communicating systems in the 90's and finally fully integrated, higly interconnected systems executing distributed control algorithms for the next vehicle generation. In order to intelligently manage this evolution, the Mercedes-Benz (MB) strategy for electronic vehicle system design consists of four major aspects:

1. minimization of cost
2. ability to manage complexity
3. maximization of reliability
4. optimization of functionality.

This strategy mandates a systematic design process which takes into account the workshare between manufacurer and supplier. From the point of view of the car manufacturer, system design is the essential key to an intelligent management of a design process which is heavily restricted by time-to-market, quality and cost aspects as well as the recognition of competitive advantages. Therefore, a car manufacurer has to have as much information about a vehicle system as is necessary to intelligently decide about system architecture and development depth of the components. Last but not least, system integration test as well as the preparation of diagnostics and service information has to be done under the responsibility of the car manufacturer.

Traditional design cycles are characterized by a number of aspects:

- nonformal descriptions of requirements and design results
- heterogeneous and inconsistent design data
- long iteration cycles, typically mandating vehicle validation tests and supplier involvement.

Currently, new design methodologies provide the basis for the optimized execution of existing and future vehicle development projects. Major components are:

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00

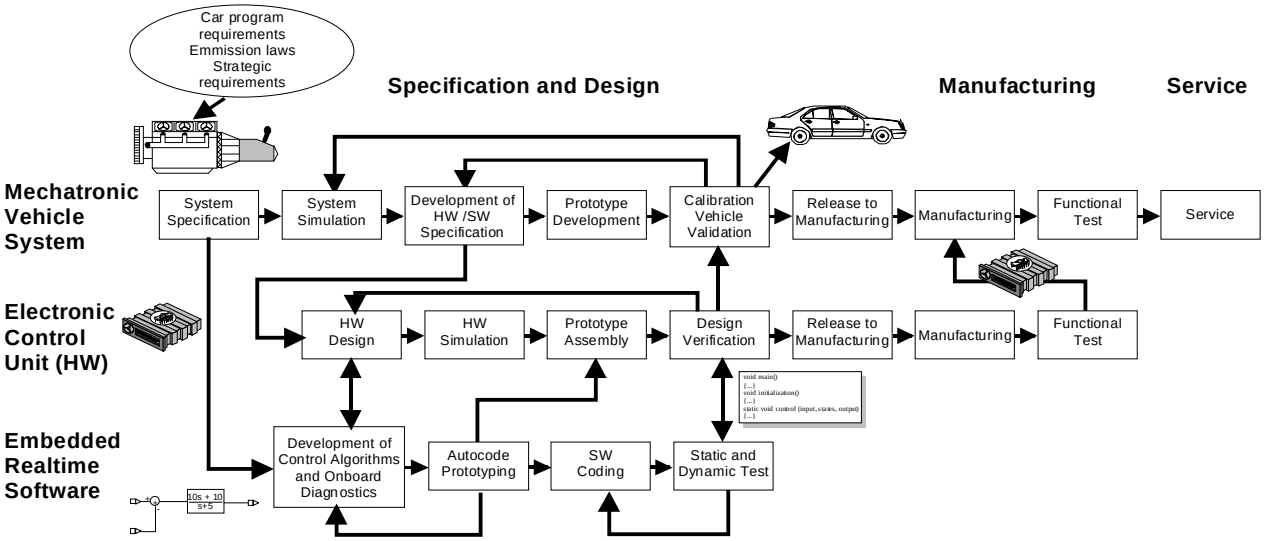

Figure 1. Concurrent development processes for mechatronic vehicle systems

- Process reengineering and measurement based quality improvement methods
- Sophisticated design data and documentation management
- Increasing formalization of specification and design descriptions
- Multi-level, mixed-mode system simulation under real-time constraints including mechanical, hydraulical, pneumatical, electrical and electronical parts
- Continuous support for specification, design, rapid prototyping, test, manufacturing and service.

For automotive development processes, it is very important to take into consideration the interaction between manufacturer and supplier. As described by Clark and Fujimoto in [1], three major scenarios have to be distinguished (figure 2):

- Detailed development of the ECUs is performed by the supplier companies. The manufacturer selects components based on his overal car concept and due to cost, performance or strategic aspects. The know-how is mainly avalaible at the supplier's organization.

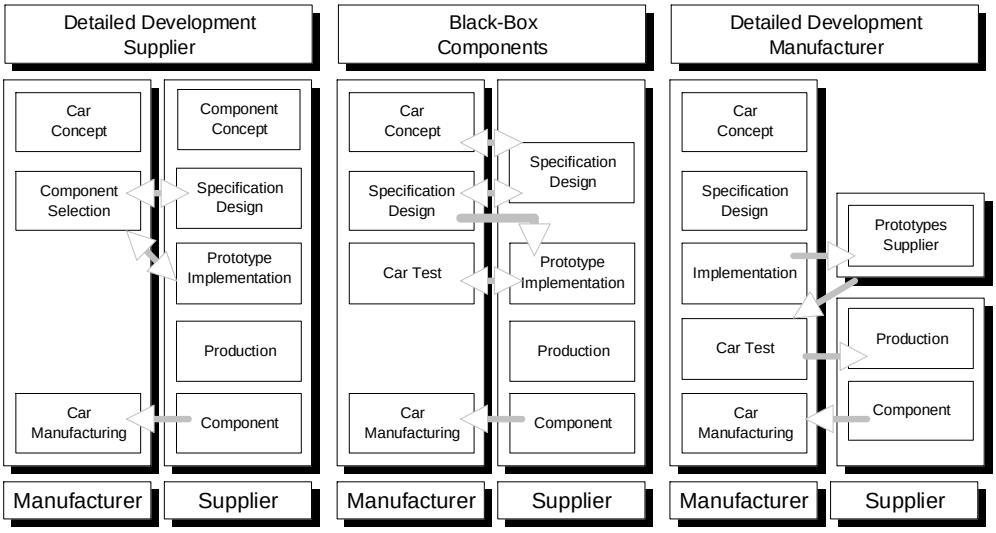

Figure 2. Scenarios of manufacturer/supplier interaction

- In the case of black-box components, the manufacturer performs a detailed specification and a validation (through simulation and test) of the components, whereas the supplier implements the ECU based on these specifications. Details about the implementation are not available at the manufacturer's organization.
- Detailed development is performed by the manufacturer. The suppplier implements the prototypes based on the detailed specs or supplies a basic hardware/software level where the manufacturer implements his algorithms.

Among the numerous ECUs integrated in modern vehicles, each of these development scenarios can be found. In every case, the level of detail of the information that is exchanged has to be clearly defined. In the MSR project [2], several levels of information and product exchange are identified:

1. requirements documentation, e.g. based on SGML (Standard Generalized Markup Language)
2. specifications on different levels, FMEA (Failure Modes and Effects Analysis) /FTA (Fault Tree Analysis) data
3. prototypes (HW and SW) and functional test patterns
4. source code
5. object code
6. ECUs and related test patterns.

Regarding the hardware and software components of an ECU, one could distinguish between the scenarios illustrated in table 1. Black-box scenarios will be used for standard functions like ignition control. The grey-box and white-box scenarios are the favorite strategies for future electronic vehicle system design in the case of functions related to competitive advantages.

	Black-Box	Grey-Box	White-Box
Functional SW Modules	Supplier	Manu-facturer	Manu-facturer & SW suppliers
Operating System & I/O	Supplier	Supplier	SW supplier
Hardware	Supplier	Supplier	HW Supplier

Table 1: HW/SW interfaces for different cooperation scenarios

The following chapters will describe the development and installation of the specification and design process in more detail. Chapter 2 will describe reengineering activities as the basis for a systematic optimization of the design process. Chapter 3 will describe methods and tools used in already optimized processes. Chapter 4 will describe status and trends in electronic vehicle system architectures and Chapter 5 will provide future directions and a conclusion.

2 Process Reengineering

To avoid local optimization of design processes without taking into account the real problems in the design cycle, the introduction of new methods and tools in the MB development departments is based on a systematic reengineering and quality improvement approach. Design processes are analyzed and modelled with respect to relevant activities, related parties as well as necessary information, manpower and technology resources. Dedicated metrics and questionnaires provide support for the identification and realization of necessary process improvement. Due to the lack of reference processes for automotive system design, much effort is spend to develop a specific approach, particularily based on approaches found in [3]-[7].

The first reengineering analysis obviously showed significant potential for improvement through the introduction of system design methodologies and tools such as

1. requirements capture and guided editing as well as standard formats for requirements and design documentation
2. usage of formal, executable models as well as system simulation to detect problems early in the design cycle
3. establishing rapid prototyping the development of powerful prototyping platforms as well as code generation from executable specification models
4. systematic validation and verification activities for integration and module test as well as vehicle validation.

The introduction of formal, executable specification models showed to be the backbone of the new development process. However, the underlying methods were not established in the development departments and the tools significantly lack integration in the existing method and tool environment. The lack of an appropriate system design methodology for distributed realtime systems lead to unsatisfying results. Furthermore, introducing these approaches into the complex concurrent engineering process performed by MB and its suppliers showed to be the major problem. Due to this situation, an extended strategy is now established:

1. A major effort concentrates on adapting the methods and tools to vehicle specific aspects. This includes a model schema representing the functional and architectural aspects specific to distributed electronic vehicle systems. Furthermore, vehicle specific model libraries as well as interfaces for integrating existing functionality are developed.
2. Significant effort is spent to adapt the modelization and description languages to vehicle specific needs. A style guide is under development that includes

- a specific methodology for developing vehicle control functionality and to distribute it among an optimized architecture of electronic control units
- readability and layout rules for modelization to take into account that models are still documentation in the first place („readability before executability“)
- strictly formal semantics for language subsets used for safety critical applications such as steer-by-wire
- integration into the infomation technology process, e.g. integration with SGML-based requirements documentation including requirements traceability or integration with CAN (Controller Area Network)-specific development tools, especially bus simulation, communication matrix as well as specific estimation and optimization tools
- integration of code generation into a general realtime platform based on an scalable realtime operating system including autocode from other tools, realtime execution and communication as well as test pattern databases and data recording capabilities
- integration with diagnostics tools
- integration with man-machine interface development tools.

3. The adaption of code generation from executable models showed to be too much restricted by traditional vehicle system architectures. Therefore, new architectures including a standard realtime operating system and client/server approaches are under development.

3 Methods and Tools

Based on proprietary activities in different vehicle manufacturer and supplier companies, a joint cooperative action, the MSR project, was established to define a specification, design and test methodology that provides definition of external actuators and sensors, interfaces of the system to be designed, the internal functional structure of the system as well as the realtime behavior of the single functions. This approach was realized using an integrated environment including the tools commercial tools Statemate [8] and MatrixX [9]. Several pilot projects have been performed:

1. cruise control,
2. transmission control
3. engine management.

Based on these pilot applications, the methods and tools are currently used in several development projects:

- interconnected car body systems
- CAN gateway
- central locking systems
- active gearbox systems
- transmission control
- active suspension systems
- active braking systems.

These applications mandate the consideration of open loop and closed loop control behaviour as well as system communication. It is worth noting that analog modelling is used to represent both the internal controller behaviour and the engine and vehicle environment. In the MSR project, a prototype coupling of Statemate and MatrixX was developed to be able to simulate complex systems containing open-loop and closed-lop aspects as well as a model of the vehicle, the engine or the transmission. The integration of tools is performed at several levels:

1. Method integration: the combination of different tools mandates a disciplined approach to modelling and interfacing. In the MSR project, functional decomposition using data flow diagrams is separated from behavioral modeling using state machines, differential or difference equations, lookup tables or algebraic equations or any combination of these.
2. Tool integration: The combined simulation of different models mandates a coupling of the executing simulation engines. In the MSR project, direct coupling or the usage of simulation backplanes (like software busses) were considered. Coupling of simulators requires open interfaces as well as synchronization mechanisms.
3. Code integration: Tools providing code generation (AutoCode) can beused to integrate models on code level. Code generated from one tool can be integrated into another tool and code generated from different tools can be linked and executed on a PC, workstation or a prototype realtime platform.

The specification and design methodology is based on the following aspects:

1. continuous requirements tracing
2. integrated approaches to model-based specification of closed-loop and open-loop aspects (figure 3)
3. a systematic approach for function- vs. component oriented design
4. rapid prototyping
5. hardware-in-the-loop test.

The model based specification of the embedded realtime control functions is based on a specific methodology consisting of

- functional specification of control aspects independently from architectural aspects including usage of reusable function models
- distribution of control and datapath based on the selected architecture including generation of related communication prototcols based on the CAN standard.

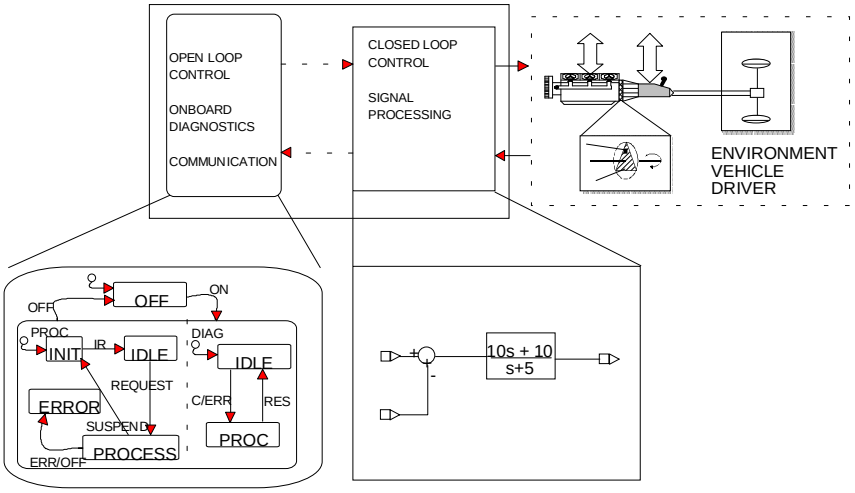

Figure 3. Integrated specification and test environment

4 System Architectures

The design of an optimized system architecture for electronic vehicle systems is influenced by function, cost, weight, packaging and placement and power consumption.

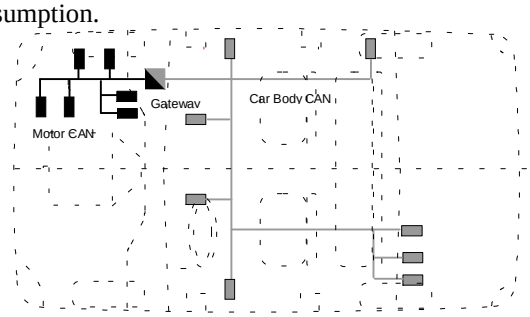

Figure 4. Current network architecture for distributed vehicle control functions

Traditional ECU development was very much influenced by the mechanical units. As a result, numerous standalone ECUs such as ignition control, injection control, or transmission control evolved. The next generation, introduced in recent car models such as the new E-class, system integration into major functional units such as complete transmission control units integrated into the packaging of the mechanical transmission component are realized. Several bus systems including high speed CAN for realtime functions like engine and transmission control and low speed CAN for car body and comfort functions are installed (fig. 4). The next generation will introduce integrated powertrain management systems followed by autonomous driving and brake-by-wire or steer-by-wire systems. A standard realtime operating system will also be part of future ECUs.

A major improvement will be based on client/server architectures (fig. 5), which allow an optimized usage of the resources available in the car. This approach enables the flexible implementation of a specified functionality on alternative constellations of processing power in the car. It is important to notice that safety critical applications like ABS and closed loop control applications mandate specific strategies in this scenario.

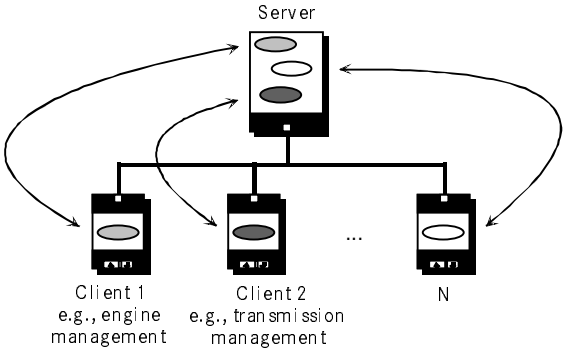

Figure 5. Client-/Server architectures for future electronic vehicle systems

Rapid prototyping is a very important aspect in the vehicle development process. Although vehicle simulation coupled with executable models of ECU specifications provide significant improvement in the early design phases, vehicle validation is absolutely necessary due to two major facts:

- the complex nonlinearities found in vehicles and its environments, e.g. combustion engines or varying road characteristics

- The feeling of the driver that can only roughly be represented by driver models.

To be able to provide a multi-level prototyping environment, powerful realtime systems consisting of PowerPC processors, realtime operating system, VME based interconnection to sensors and actuators, an existing ECU for existing functionality as well as online simulation, calibration and measurement functionality is essential for vehicle prototyping (figure 6). It is important to notice that especially the transfer of calibration data from specification to design and implementation is a major risk factor in the ECU development process.

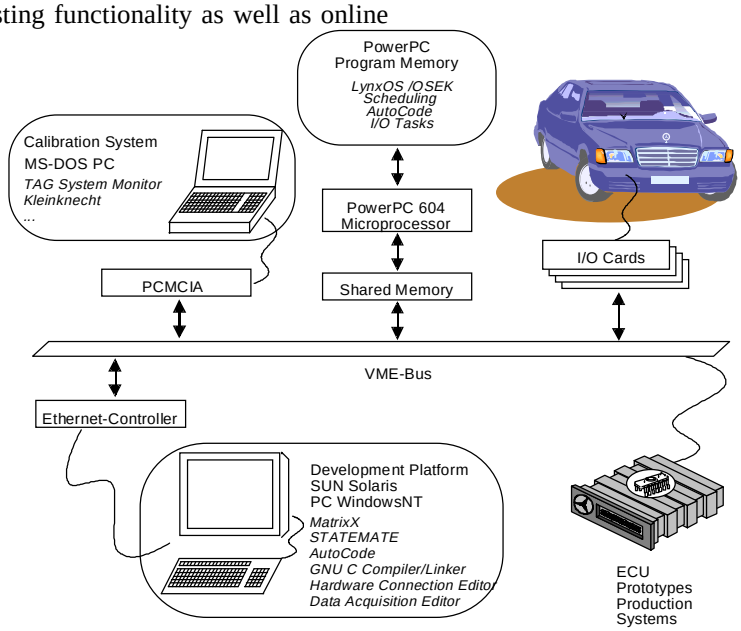

Figure 6. Integrated Rapid Prototyping environment for ECU development

5 Conclusions and Future Work

This paper gives an overview over the current activities in the area of electronic vehicle system design. The dynamic evolution of these systems with respect to growing functionality and an increased number of safety critical functions mandates new methods and tools for the development as well appropriate system architectures for prototyping and productions systems. Future applications heavily demand the following improvements:

- Library based reuse of functionality and software.
- A systematic process for design and test of safety critical applications including clear formal semantics and certification for components used in safety critical processes.
- Open interfaces for model exchange and data access for specific optimization and processing.
- Production code generation and interaction with target system instrumentation and debugging.
- Sophisticated metrics and optimization strategies for online project control.

References

[1] Clark, Fujimoto. *Product Development Performance*, Harvard Business School Press, Boston, 1991

[2] K.G. Besel , T Hirth. *Werkzeuge im MSR Projekt*, VDI Berichte Nr. 1009, 1992, pp. 503-516

[3] *ISO 9000-3: Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software*, Int'l Org. for Standardization, Geneva, 1991

[4] *ISO 9001:Quality Systems - Model for Quality Assurance in Design/Development, Production, Installation, and Servicing*, Int'l Org. for Standardization, Geneva, 1994

[5] M. Paulk e.a. *Capability Maturity Model for Software, Version 1.1*, Tech. Report CMU/SEI-93-TR-24, Software Eng. Inst., Pittsburgh 1993

[6] M. Paulk e.a. *Key Practices of the Capability Maturity Model*, Version 1.1, Tech. Report CMU/SEI-93-TR-25, Software Eng. Inst., Pittsburgh 1993

[7] *Development Guidelines For Vehicle Based Software*, MISRA Consortium, November 1994.

[8] Statemate, Product of I-Logix Inc., Boston, Massachussetts

[9] MatrixX, Product of Integrated Systems Inc., Santa Clara, California