Code vs. Cable
Model-based design and evaluation of E/E systems

Dipl.-Ing.(FH) Jürgen Kaiser , Gigatronik Austria, Graz/Austria

Dr. Klaus Baron, Delphi body & safety, Wuppertal/Germany

Jeff Ehlers, MotoTron, Oshkosh/USA

Abstract

This paper presents a complete, model-based methodology for the design, optimization and evaluation of E/E systems in vehicles. The E/E system is first subdivided into functions which are in turn resolved into engineering objects (HW, SW, sensors/actuators, electrical). The functional models can be generated from existing circuit diagrams, Simulink models and C code. The links between functions and dependencies between and within functions are modeled graphically. The functions are depicted using the vehicle topology as a basis, and the optimized harness (costs, weight, splices, ground points, …) is calculated. The structure of the controllers and bus systems is then derived from the topology, whilst communications (messages, gateway tables, send/receive logic) is generated automatically (CAN, LIN, *Flexray*). Here the ECU resources (RAM, ROM, flash, performance, tasks, space requirement, heat-up,….) are optimized according to the distribution of functions.

The controller software is generated from the functional logic on the basis of the model, the code being generated automatically and loaded into the real controller. The controller / complete system documentation is automatically generated according to the variant.

The entire E/E system can be subjected to a (top-down) failure analysis (cables, ECUs, SW, signals). The causes of failures can be analyzed (bottom-up). The effects of modifications can be demonstrated and evaluated.

The paper presents the integration of all the necessary tools on a single Engineering Data Backbone. The exchange of data with the Backbone and associated cooperation between the tools takes place via Services and uses well-known standards (e.g. FIBEX, XML, AUTOSAR). This offers a transparent, standards-based solution.

1. Motivation

The development of Electrical/Electronic architectures for vehicles is rapidly becoming ever more important. Factors important to the success of a vehicle, such as cost, quality and operational reliability, are strongly influenced by electronic network architecture and the layout of its components. Whilst the vehicle electrical system of the early 90s was still the domain of electrical engineers and power distribution specialists, today it has turned into complex communication system with several different bus systems and gateways. Nowadays, functionality is no longer implemented in stand-alone ECUs, but rather and to an increasing extent distributed among several controllers, in order to conserve resources and save on components. This process will intensify as a result of forthcoming functions such as driver assistance and active safety systems.

A glimpse into the future of hybrid and purely electrical vehicles with wheel hub motors illustrates how demands on the E/E system can be expected to escalate further with regard to reliability and safety. The level of complexity does not rise because of the inventiveness of engineers but rather solely as a result of increasing functionality. Ever more frequently, suppliers find themselves "confronted" by OEM requests to offer complete E/E systems that not only should fulfill ever rising demands for quality and operational reliability but also must be produced and maintained at ever lower cost. Suppliers are therefore obliged to build up in a very short time all of the necessary capabilities for designing and implementing E/E systems. Customer requirements and inhouse designs must be evaluated with a high level of confidence in order to avoid costly experimentation. This includes not only the capability to design and evaluate complete E/E architectures but also the implementation of all components, from the harness through to the controllers and their software. Here the forthcoming AUTOSAR standards must also be taken into consideration.

These radical changes, coupled with high levels of investment and personnel costs, compel suppliers to devote massive effort to rebuilding their engineering processes, without which no one will be in a position to make a competitive bid in future. Transition to virtual system design and model-based development will no longer be an option but rather the key to survival. Without new methods and tools that enable a high degree of automation of the system optimization process and evaluation of results, the necessary quantum leap will not be achievable.

This paper describes an appropriate approach to integrated design of E/E systems in vehicles through the integration of associated engineering disciplines and tools, the level of implementation achieved and a glimpse into the near future.

2. Functional Design

The most important step in the transition to virtual design and model-based development is the separation of solution and implementation. Instead of immediately thinking in terms of components and software modules, first an abstract, reusable model of the engineers' proposed solution ("virtual solution") is generated. This model fulfills the required functionality and can be evaluated accordingly. However, the production of a virtual solution of a complete E/E system requires models that span several engineering disciplines: The design and evaluation of the vehicle electrical system (harness, connecters, fusing concept, power distribution, energy management, …) including the factors of cost, weight, packaging, fitting, ease of maintenance.

The design and configuration of the communication network (bus systems, protocols, configuration of messages, frames, gateway configuration, …) including the factors of security of transmission, latency times, wakeup concept. Software engineering implementation of networked functions that are realized distributed across several controllers. When it comes to the integration of different engineering disciplines, the next problem is definition of the term "function". An electrical engineer interprets it somewhat differently from a software developer, with the corresponding tools and models also being incompatible with one another. Yet the modeling and evaluation of a complete E/E system makes a holistic, virtual rendering of all system elements indispensable. In the approach described here, three modeling languages are integrated into a holistic approach:

- virtual modeling aided by "Virtual Solution Design" using **ESCAPE**
- functional modeling by the harness optimization tool **eSCOUT**
- MATLAB-Simulink-based ECU SW modeling with **MotoHawk**

2.1 Functions in the electrical network

Here we consider that part of the vehicle electrical system realized in hardware. Functions are defined as solution loops running from the sensor to the I/O channel (e.g. digital input) on the controller and from the I/O channel (e.g. PWM output) to the actuator. Figure 1 shows an example of the function "rear window defroster" with a switch for rear window defroster having an integrated LED to confirm when heating is active, the heating element itself and the associated I/O channels.

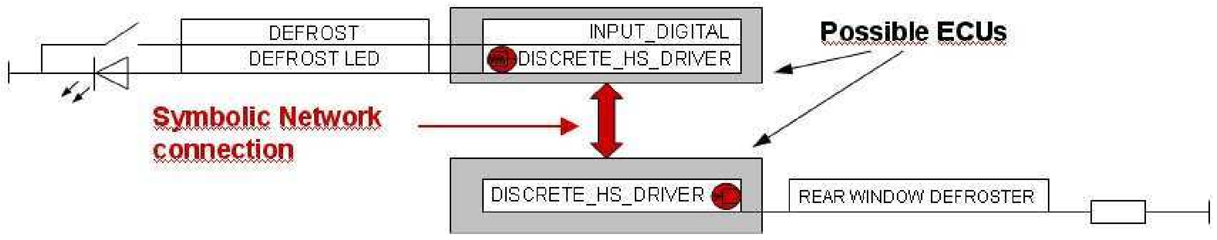


Figure 1: "Rear window defroster" function, electrical

The electrical components for one function are assembled as base model objects and managed in a component database, with the rear window defroster switch consisting of two partial objects, switch and LED for example (see Figure 2). Along with the model objects, all necessary data are stored for later optimization and evaluation of the E/E architecture and its components, e.g. space requirement for I/O switching on the circuit board, power consumption, power losses.

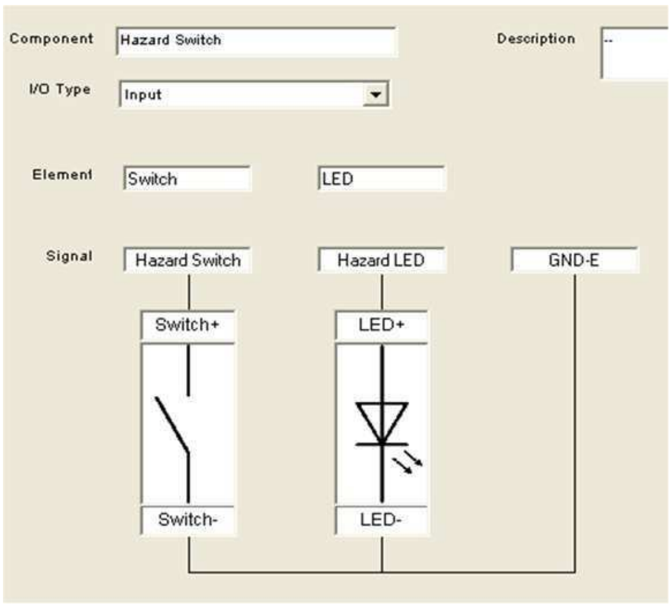


Figure 2: Electrical components in eSCOUT

2.2 Functions in the functional network

A function is depicted as a complete solution in the functional network. For this a so-called "Virtual Solution Design" is used, representing the process sequence from sensor to actuator. With this method, using three basic elements every form of electronic device can be modeled, from a coffee machine to an Airbus A380.

The three basic elements are:

- **Process types:** execute a transformation of process data (e.g. temperature, pressure, lateral acceleration) into electrical values (current, voltage) -> sensors or, in the reverse direction, electrical values into process influence (actuators)
- **Hardware types:** execute a transformation of electrical currents and voltages into digital data (-> input channels) or of digital data into electrical currents and voltages (-> output channels)
- **Software types:** execute an action that processes digital input values and delivers digital output values

Each of these types possesses other data that are required for the optimization and evaluation of an E/E architecture. For all three elements it is possible to define dependencies between inputs and outputs for a subsequent analysis of the risk of failure and to store corresponding simulation models, e.g. MATLAB/Simulink. The process type and hardware type basic elements are identical to the basic elements of the vehicle electrical system design and can be generated automatically from it. The software type processes digital information and converts it into new information. For example: a software function that connects two binary inputs via an AND function delivers information to its output, shifting from condition “0” to condition “1” if all of the inputs carry the value “1”.

If the individual model elements are linked together, “connections” arise. Connections represent the flow of information from one model element to one or several others. The information to be transferred consists of the form (data type) and content (data). If two model elements linked by connections are distributed over several ECUs, connections become “signals”. However, in order to be able to reuse solutions, the functions and their model elements must be viable without connections or signals, given that these only arise when one connects two model elements (functional building blocks). Each fundamental building block therefore needs input and output elements that contain the form of the information that is needed or delivered. The content is only created after simulation or trial in a real electronic system, e.g. form: binary, content: 0 or 1.

Terminals – inputs & outputs as data objects

In Virtual Solution Design, the input and output elements are called “terminals”, because attachment occurs there. Terminals are object types that describe the interface of a function or a basic element. The description of the form of data does not only contain the data type. Value ranges and their significance must also be known in order to be able to link inputs and outputs consistently. In the world of electronics a value of “0” or “1” is not yet indicative of the actual process condition. Depending on the switching specification of hardware type, a “0” may indicate “on” or “off”. As with the fundamental building blocks, the terminals also have three basic types:

- **Process terminals:**

  input/output for the process; data types: pressure, temperature, rotation speed,….

- **Hardware terminals:**

  electrical (analog) input/output; data types: current, voltage

- **Software terminals:**

  digital input/output; data types: BOOL, USINT, ….

The fundamental building blocks are used to represent a mechatronic system in the form of a process sequence. If simulation models are stored for the individual types, the entire process sequence can be simulated. The electrical components of a process sequence can be taken automatically out of the electrical circuit diagram (electrical functional building block) and carried over to a virtual solution model. Here the model elements for the application software must be inserted manually. If one reverses the procedure and defines the virtual solution model directly, deriving the electrical circuit diagram from it, then the task of software definition can be dispensed with because the software is already part of the solution model. Figure 3 shows the solution model for the “rear window defroster” example from Figure 1.

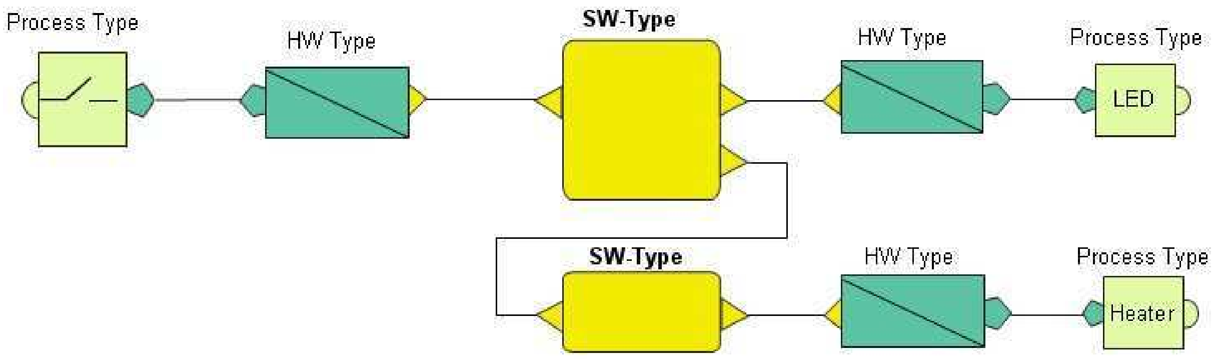


Figure 3: Virtual solution design model for rear window defroster

3. Optimization of hardware deployment

The locations for installation of sensors and actuators, as well as potential spaces for locating controllers and laying cables, are defined in a topological layout (see Figure 4). This information will subsequently be required by the optimization algorithm in order to place the model elements within the vehicle.

A great deal of information pertaining to location will already have been processed and predetermined by other disciplines. The packaging spaces for lamps, switches, interior lighting and connection interfaces between sub-harnesses, e.g. for the door, are already defined during mechanical design, with the electrical engineer having little or no influence on them. Thus the room for maneuver during optimization of the electrical system is severely limited. More involved optimizations, such as relocation of the battery from the engine bay to the trunk for example, require consideration of relocation of fuses, e.g. relocation of the fuse box to the trunk or splitting the fuse box into 2 or more smaller fuse boxes.

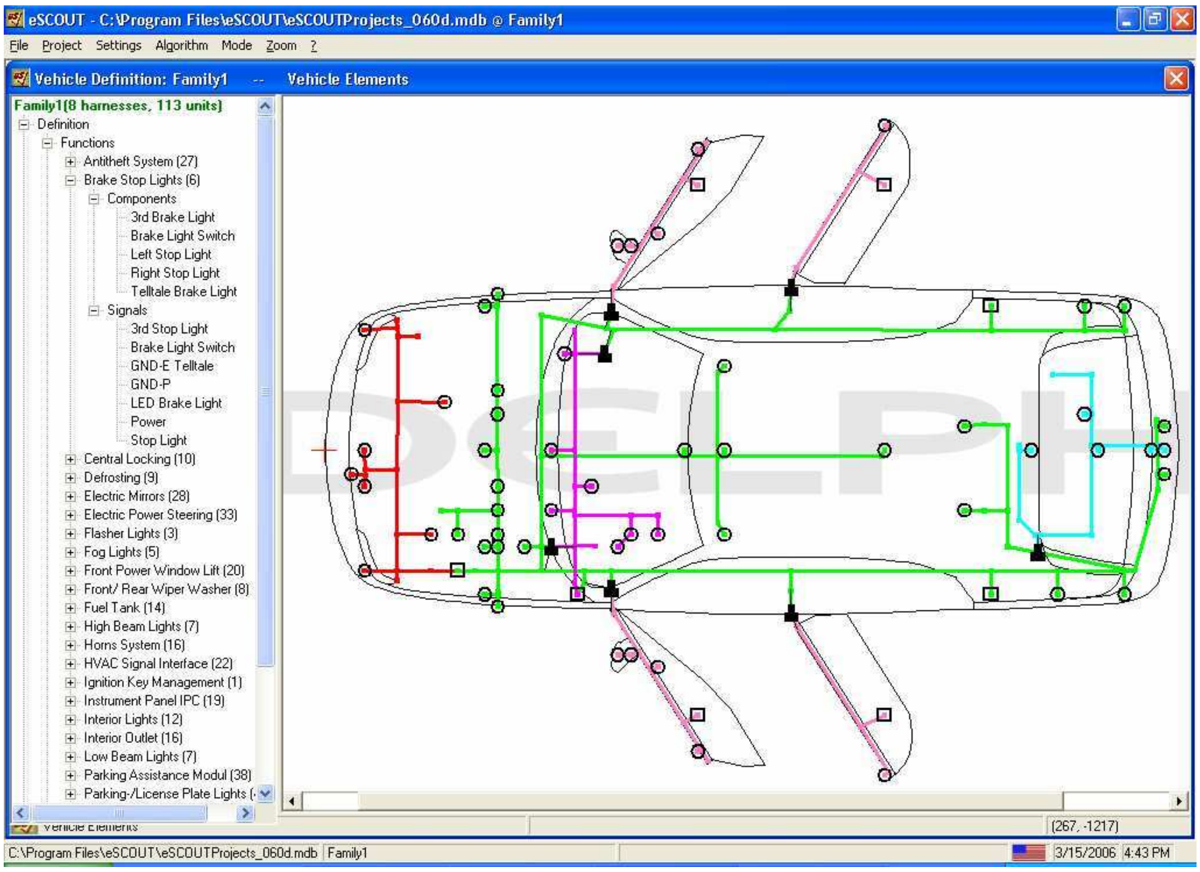


Figure 4: Definition of topology data

Such considerations are normally the reserve of the OEM because of the massive influence they can have on vehicle design and perception by the ultimate customer, the driver. The first step in optimization is therefore definition of the architecture, i.e. number of controllers, use of “smart components” (e.g. LIN nodes), bus connections and gateways. Figure 5 illustrates 4 possible architectures with corresponding distribution of I/O channels to the controllers.

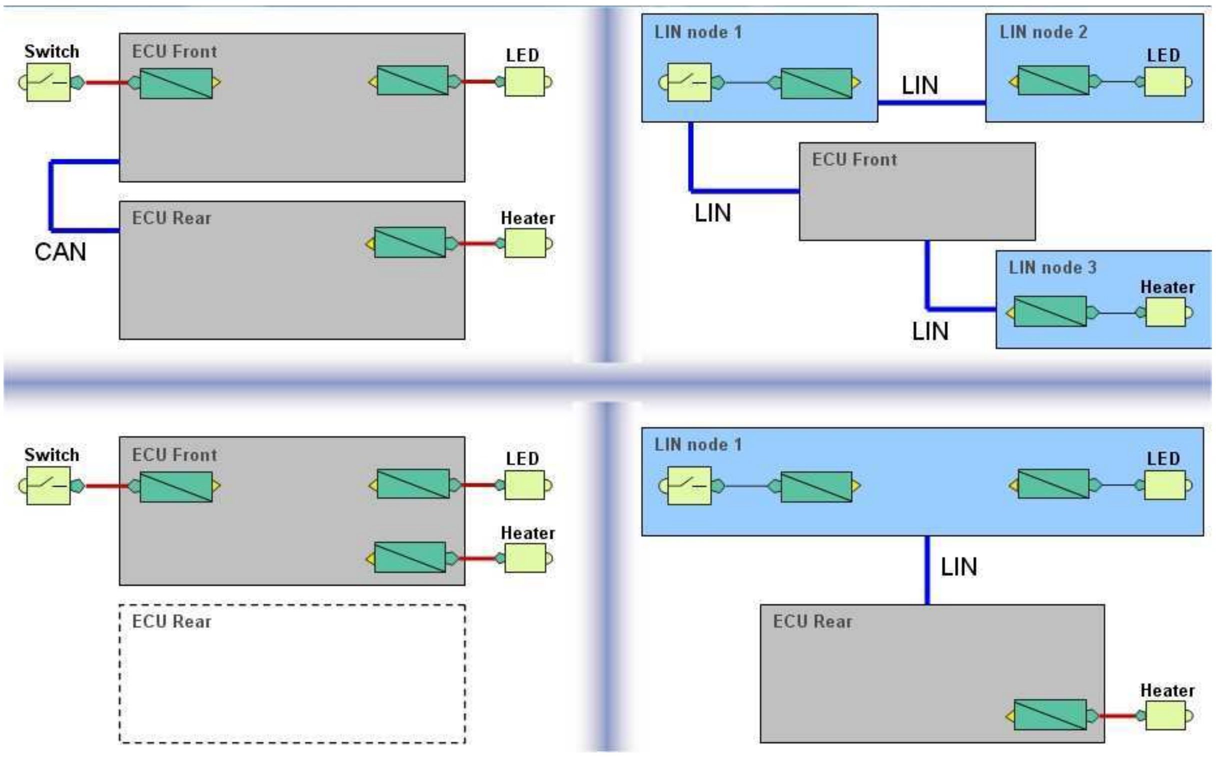
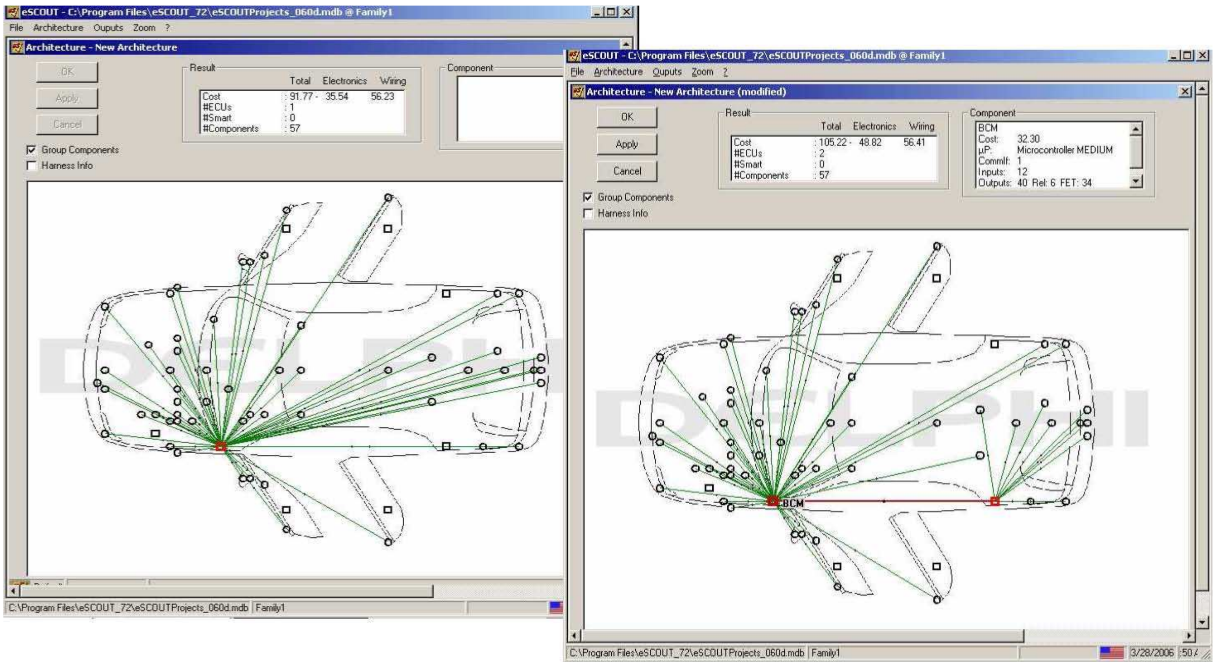


Figure 5: Possible architectures, examples of hardware mapping

These structures can be created with the aid of ESCAPE or eSCOUT. Here eSCOUT automatically distributes I/O channels (HW types) among controllers and works out the corresponding harness. If the resulting architecture does not fulfill the requirements, another structure is created and worked out, the process repeating till the termination criterion is reached, namely architecture fulfills requirements. Figure 6 illustrates two possible architectures for the body zone of a vehicle with one or two controllers. If one uses 2 controllers (ECU Front and ECU Rear), the cabling requirement reduces because sensors and actuators in the rear part of the vehicle are connected to the rear controller. Disadvantage of the solution: data must be distributed via a bus system. This implies additional effort in the area of software, with regard to both software resources (processor power required) and hardware resources (RAM, ROM, flash memory).


eSCOUT - C:\Program Files\eSCOUT_72\eSCOUTProjects_060d.mdb @ Family1
File Architecture Ouputs Zoom ?
Architecture - New Architecture
OK
Apply
Cancel
Group Components
Harness Info
C:\Program Files\eSCOUT_72\eSCOUTProjects_060d.mdb Family1
eSCOUT - C:\Program Files\eSCOUT_72\eSCOUTProjects_060d.mdb @ Family1
File Architecture Ouputs Zoom ?
Architecture - New Architecture (modified)
OK
Apply
Cancel
Group Components
Harness Info
BCM
C:\Program Files\eSCOUT_72\eSCOUTProjects_060d.mdb Family1
3/28/2006

Result

	Total	Electronics	Wiring
Cost	: 91.77 -	35.54	56.23
#ECUs	: 1		
#Smart	: 0		
#Components	: 57		

Result

	Total	Electronics	Wiring
Cost	: 105.22 -	48.82	56.41
#ECUs	: 2		
#Smart	: 0		
#Components	: 57		

Component

```
BCM
Cost:     32.30
µP:       Microcontroller MEDIUM
CommIf:   1
Inputs:   12
Outputs:  40  Rel: 6  FET: 34
```

Figure 6: Harness optimization by means of eSCOUT

So optimization of the electrical aspect through reduction of cabling leads to increased effort on the electronic side.

4. Optimization of software distribution

The next step therefore involves an optimization of the distribution of software amongst controllers. This takes account of both optimization of the use of available processor resources (demanded power available, lowest cost of microcontroller) and optimization of bus loading. Bus communication effort will be increased or decreased depending on the distribution of the software-types. By way of example, an architecture with two controllers is proposed, onto which the solution model for rear window defroster is to be mapped. Figure 7 shows 4 possible solutions for software distribution. Either 1, 2 or 4 signals have to be transferred depending on the distribution of software components

5. Failure Risk Analysis

A failure mode and effect analysis can be carried out within the solution model. Figure 7 illustrates a supposed failure of the “ECU Rear” controller along with the consequences for this case. The model elements marked red indicate a primary failure (because they are mapped to the ECU Rear controller), whilst the orange-colored elements indicate a secondary failure (data from failed system components is missing). For the example shown, in two cases of distribution an indication of rear window defroster failure could be achieved through a blinking LED, whilst in the other two cases this would not be possible.

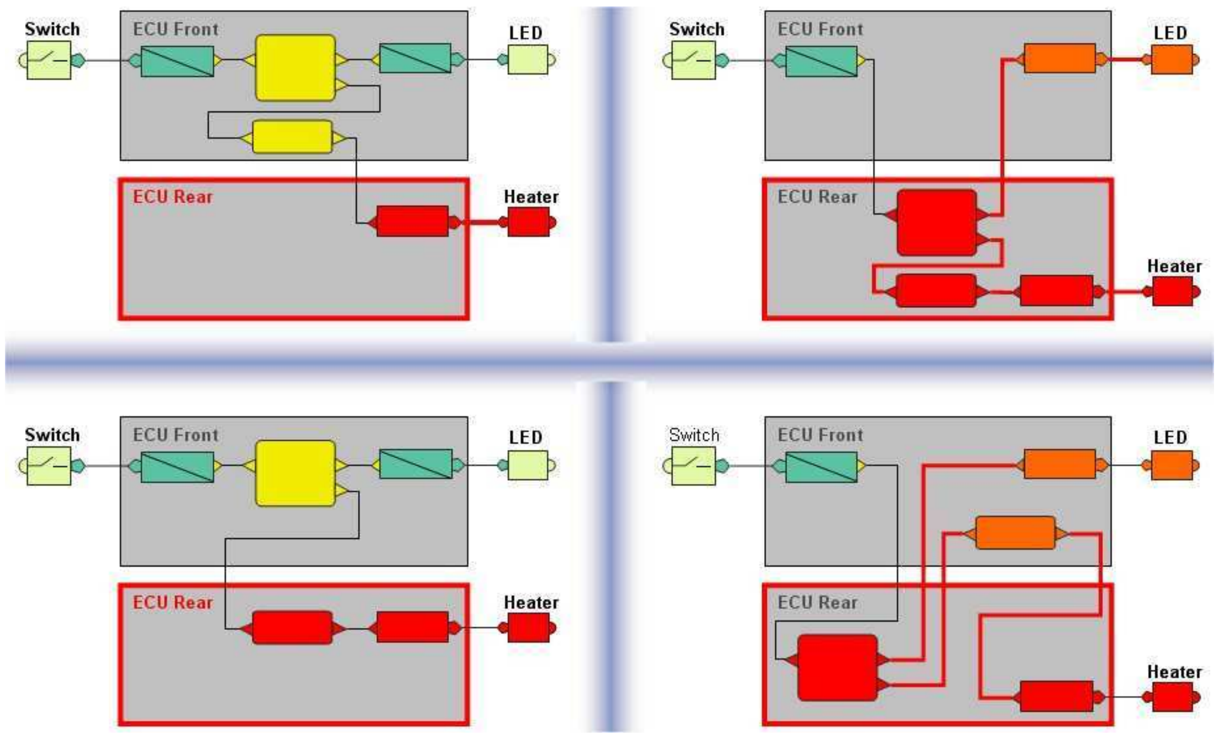


Figure 7: Software mapping and failure risk analysis

The model element dependency matrices used for the failure risk analysis can also be used for failure analysis and diagnosis. For this, the failed element is selected and the dependency analysis carried out in the reverse direction.

6. Optimization of latency times / automatic bus configuration

Optimization of the distribution can become very complex. Software distribution around a network can, in some circumstances, substantially increase the latency time of a signal from switch operation till activation of the actuator. In the case of rear window defroster, a prolonged reaction time is no great cause for concern. However, when switching electrically operated windows, it is indeed annoying if the motor does not react immediately and one has already selected the opposite direction before the actuator has responded.

This becomes safety-critical for example in the case of the brake pedal sensor which delivers the data for activating the brake light. Here every millisecond counts, given the effect on the reaction time of the person behind, potentially making the difference between a crash and timely braking. As shown in Figure 8, the calculation of latency time requires consideration of all elements in a process sequence.

In the case of simple hardware logic combined with manual software implementation, accessing of the hardware consists of simple bit commands from a microcontroller for example whilst, in the case of complex assemblies such as those using AUTOSAR-RTE or other real time operating systems, it may involve in some cases the calling up of services from the operating system, communication or hardware abstraction levels with associated task swaps and stack management, requiring correspondingly increased execution times. This must be taken into account when calculating the worst case execution time.

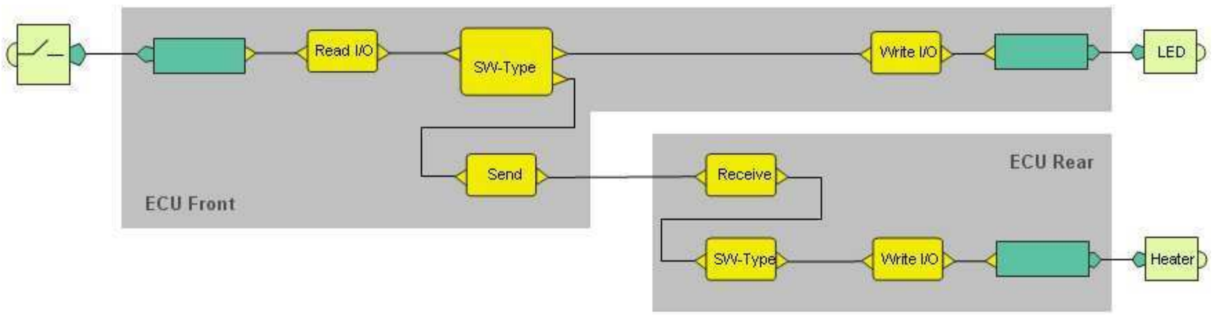


Figure 8: Model of rear window defroster with I/O access and bus logic

The reaction time (latency time) of the system is influenced not only by the distribution of software elements among controllers but also by the configuration of bus systems and gateways. Optimization of an architecture must therefore also include optimization of communication via all bus systems and gateways. The CAPEopticon tool ensures this by carrying out an automatic bus configuration.

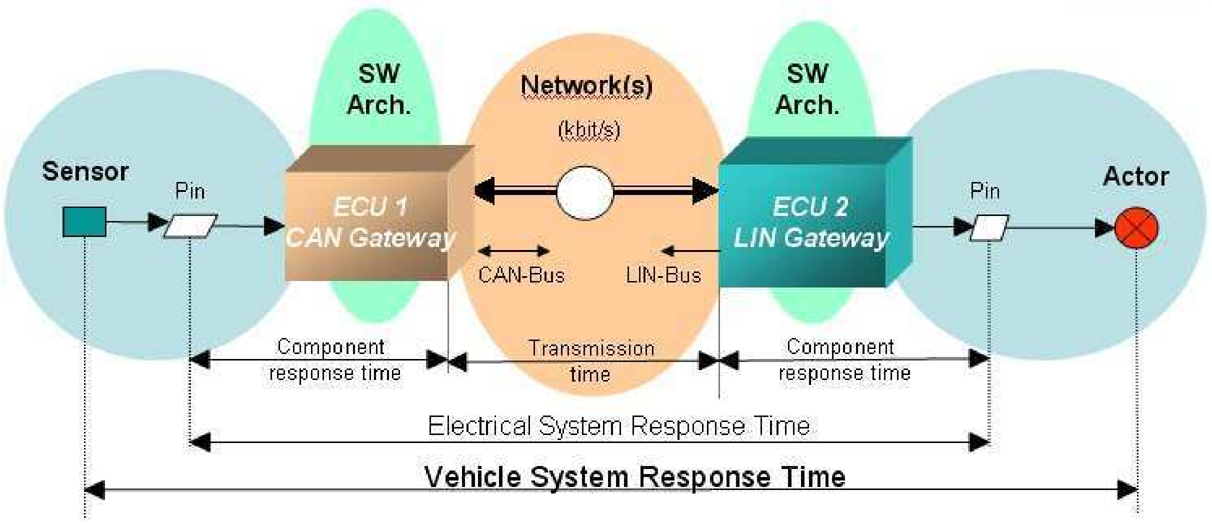


Figure 9: Response time calculation

Currently both CAN and LIN are already supported, whilst further bus systems are in the pipeline. Automatic bus configuration takes temporal constraints into account, i.e. critical signals are configured with higher priority (CAN) or shorter timeslots (LIN) for example. The gateways then handle such signals appropriately.

Figure 10 illustrates what is actually a simple problem: a sensor is read, a controller takes on the processing and controls an actuator. In the simplest case a switch that turns the rear window defroster on. However, in the example shown, both sensor and actuator are LIN components, the processing software runs in an ECU on the CAN bus and connected to the LIN components via a CAN-LIN-gateway. In this case a total of 13 objects are required from sensor through to actuator, each introducing a response delay. In addition, further delays are possible if, for example, CAN bus messages with higher priority hinder the transmission of information.

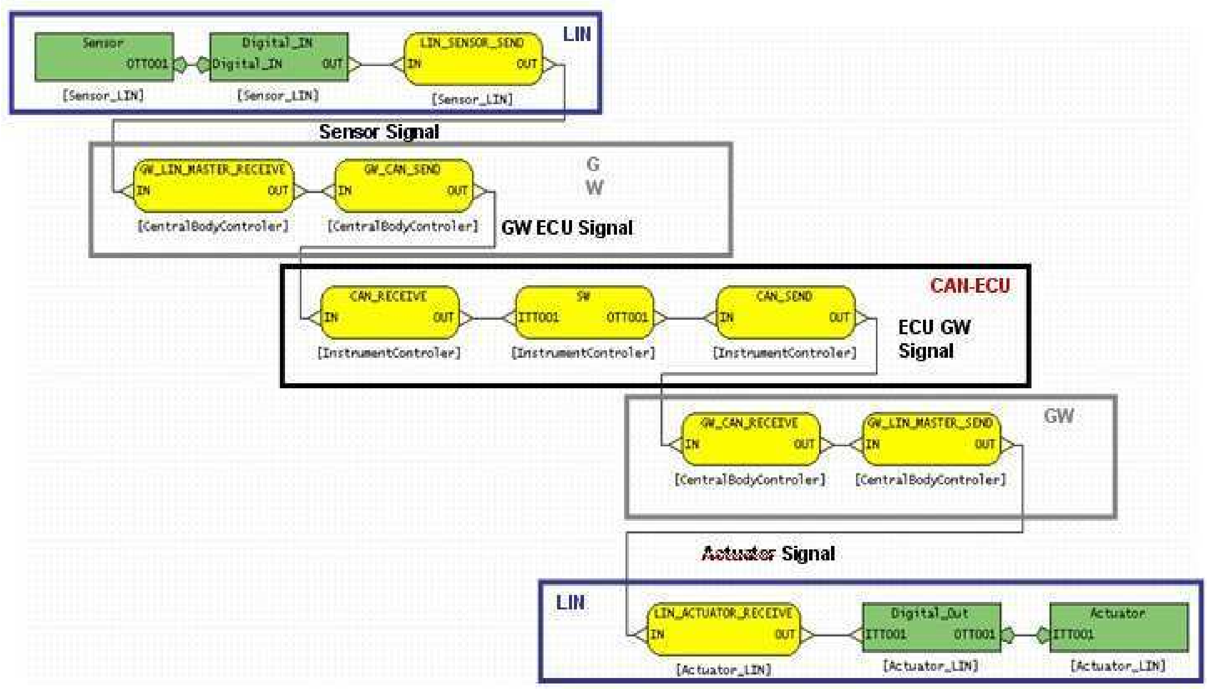


Figure 10: Example with CAN-LIN gateway in the signal pathway

Optimization of the signal pathway is accomplished with the aid of the CAPEopticon tool. By means of the model, CAPEopticon knows all of the signal's stations and attempts to find the optimum path and optimal configuration taking into account all of the signals potentially to be transmitted. The data are packaged into messages and, in the case of CAN, are assigned priorities. After the optimization, the level of software effort for transmission and associated resource consumption along with bus loading on the communication systems involved are all known and can be accessed for evaluation. The results are exchanged between active tools in FIBEX format.

7. Software generation and resource calculations

A Simulink model (and/or C code implementation) can be provided for each SW-type in the type library. If such a type entity is generated, the corresponding Simulink model is then used with it. Models, assembled from many individual elements to yield a larger solution, can be automatically converted into Simulink models. ESCAPE generates these models automatically depending on user selection. If solution models and functions with underlying Simulink models are mapped onto a controller, the controller simulation can be generated automatically. The application of reusable software types from a library not only raises efficiency and quality but also allows resource consumption to be defined exactly. As well as supporting MATLAB-Simulink, ESCAPE particularly supports the Simulink-based MotoHawk development platform. Thorough application of all these integrated tools allows an extremely efficient development process to be built up (see Figure 11).

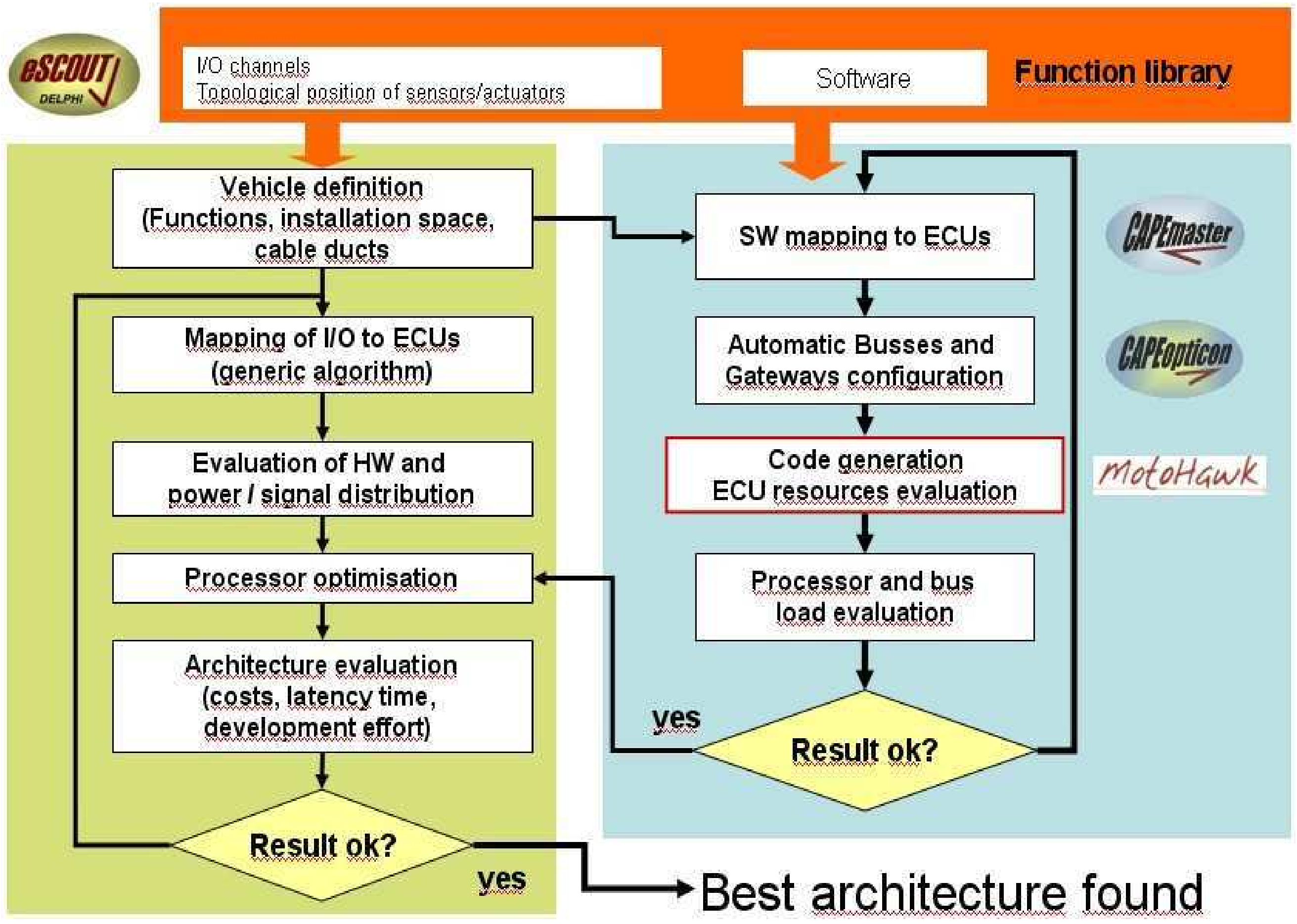

eSCOUT DELPHI
I/O channels
Topological position of sensors/actuators
Software
Function library
Vehicle definition (Functions, installation space, cable ducts
Mapping of I/O to ECUs (generic algorithm)
Evaluation of HW and power / signal distribution
Processor optimisation
Architecture evaluation (costs, latency time, development effort)
Result ok?
yes
SW mapping to ECUs
Automatic Busses and Gateways configuration
Code generation ECU resources evaluation
Processor and bus load evaluation
Result ok?
yes
CAPEmaster
CAPEopticon
MotoHawk
Best architecture found

Figure 11: Integrated design and optimization process

After automatic bus configuration, the necessary bus logic can be generated automatically by

CAPEopticon. Thus all resources necessary for communication are also known exactly. After processor optimization, the target configuration can also be defined automatically, to the extent permitted by the degree of hardware standardization. The last link in the chain still outstanding is the automatic generation of task mapping from the functional requirements. The ultimate goal ofi ntegrated design is the completel yautomatic generation of software code for all controllers in the network after designing and evaluating an architecture. In this way, changes at a late phase of the project become less of a concern, and future products can be evermore easily individualized.

8. Generation of documentation

The application of virtual solution models carries with it the further advantage that all of the information relevant to the solution elements is included. After the mapping of functions onto controllers, the documentation and requirement specification for a controller can be generated automatically via XML output and XSLT.

9.Engineering Data Backbone

Until now, it was usual to attempt to linkup all of thenecessary tools into a so-called “tool chain”. This inflexible type of coupling involves a large effort in preparation and maintenance.

Ever more complex E/E systems and numerous variants force data management to be open to all kinds of tools and retain control over any level of complexity. For this reason, the integrated engineering approach involves tools no longer being directly coupled together. Instead they operate on shared engineering objects that are managed via a so-called Engineering Data Backbone. The various tools share the same objects and deposit their respective data there. They may require these data themselves or have to provide them to other tools or users (see Figure 12). Each engineering object (e.g. ECU, software type, requirement,…) has a unique ID, allowing access to its data. A multi-user handling system protects the objects from unauthorized access and enables cross-border collaboration between engineering teams.

10. AUTOSAR

The introduction of standards such as AUTOSAR is an important step forward in terms of raising efficiency in the development of E/E systems in vehicles. If one looks at a function from the point of view of virtual solution design, the introduction of AUTOSAR only changes the mapping of the solution onto the structure of AUTOSAR-RTE. Furthermore, the solution model offers the choice as to whether the solution should be implemented with or without AUTOSAR or indeed the application of AUTOSAR components and conventional controllers in the same network. Figure 13 shows the mapping of model elements on AUTOSAR.

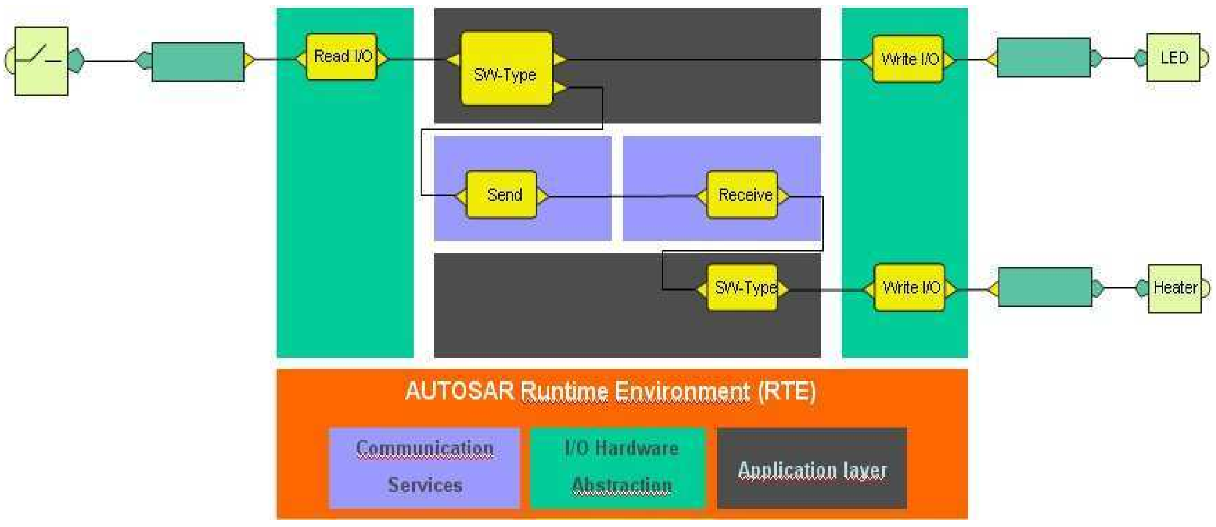


Figure 13: Solution models mapped onto AUTOSAR components

11. Summary

Given the current state of the art and the tools now available, fully integrated engineering is no longer some vision of the future. The tools for the design and evaluation of vehicle E/E architectures are by and large already available, with the missing building blocks able to be realized through manageable outlay. Without such integration, the challenges of the immediate future will not be overcome. Here AUTOSAR is no contradiction but rather an important, fundamental prerequisite.

References:

1 Kaiser, J.: „Functional modelling and design of automotive control systems“, Conference „Electronics in automobiles“, Ludwigsburg 2005

2 Dr.Baron, Klaus, „Virtual system development – From requirements to control units“, Conference „Virtual Vehicle Creation“, Stuttgart 2007

3 B.Chandrasekaran, A.K.Goel, Y.Iwasaki: „Functional Representation as Design Rationale“ IEEE Computer, Vol 26 No.1,S.28-37, 1993

4 Bracklo, C: Flexray – „Standardization in the automotive industry“; Conference „Data communication in automobiles“, Heidelberg März 2004

5 Kaiser J. et al, „Integrated Computer Aided Project Engineering“, World Congress for Railway Research, 1997, Florenz

6 Kaiser,J., Lehr, Markus: „Optimization of E/E-architectures in vehicles using model based development of mechatronics“, 4.Workshop Mechatronics, Paderborn, 2006

7 Kaiser,J.: „PDM as Engineering Data Backbone for the development of mechatronic systems“, Conference „Product Life Live“, Mainz 2006

8 Kentaro Yoshimura: „A Dependable E/E Architecture for X-By-Wire Systems Based on Autonomous Decentralized Concept“, Hitachi Automotive Systems Europe GmbH , Conference „Electronics in automobiles, Baden Baden Oktober 2005

9 McKinsey, strategy analysis of E/E systems, 2002

10 Reichart, G: „Future system architectures in automobiles“; in: 25 Jahre Elektronik-Systeme im Kraftfahrzeug (B. Bäker, ed.), Haus der Technik Fachbuch 50, Juli 2005, pp 13- 27.

11 Reichart, G.: LIN-Subbus-Standard integriert in offene Systemarchitektur, BMW AG München, , First International LIN-Conference, Ludwigsburg, Sept. 2002

12 Suad Kajtazovic, Christian Steger, Andreas Schuhai, and Markus Pistauer. „Automatic generation of a verification platform for heterogeneous system designs“. *In Advances in Design and Specification Languages for SoCs -Selected Contributions from FDL'05,* Vachoux, Alain (Ed.), Kluwer Academic Publishers Boston/Dordrecht/London, 2006

13 L. Wang, J. Wang, and I. Hagiwara, “Modeling approach of functional model for multidomain system”, JSME Int. Journal, Series C, vol.48, no.1, pp. 70-80, 2005.

14 J. Axelsson, “Holistic object-oriented modeling of distributed automotive real-time control applications”, Second IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 1999, pp. 85 – 92.

15 L. Horvath, I. J. Rudas, “Modeling behavior of engineering objects using design intent model”, 29th annual Conference of the IEEE Industrial Electronics Society (IECON’03), vol.1, pp. 872-876, November 2-6, 2003.