

Technology for a New Automotive Era

- ✓ Growing stringent environmental and safety regulations
- ✓ End-user comfort and entertainment value
- ✓ Increased pressure on cost and differentiation factors

These are the main forces propelling the automotive industry into a complete new technological world: the electronic revolution.

According to recent market studies, the embedded electronic value per car will increase from \$900 today up to \$2,000 in the next decade⁽¹⁾, which translates into an increase from 20 to 100 Electronic Control Units (ECUs) per car. Tomorrow's car will be a complex embedded electronic system with an embedded network interconnecting these dozens of ECUs and communicating with the outside world using GPS-based and wireless technologies.

Although the automotive industry has started to shift from the mechanical era to an electronic paradigm, it is still in infancy regarding complex software development practices. Developing tomorrow's automotive software is one of the biggest challenges ever known, since it has to match safety critical, distributed and cost-effective constraints, all at the same time. This challenge is magnified by the estimated size of future software developments. One of the world's largest manufacturers, Ford, predicts that there will be an increase of five million lines of code in the next five years (a 50% increase)⁽²⁾. And that is only for its Engine Management System (EMS) software!

Today, no supplier can offer an integrated approach that can cope with all automotive software requirements. For example, two different but integrated behavioral paradigms – event-driven and time-triggered real-time applications – have to be addressed. However, other industries have already successfully pushed beyond some limits using object-oriented and formal techniques. There are existing and mature software development technologies already widely used in the telecom and avionics industries that can help the automotive industry achieve significant improvements as it enters this new era.

DRIVING FORCES

Regulations

New pollution regulations make Engine Management Systems (EMS) essential for pollution control and fuel consumption while bringing engine performance to a higher level. Other safety regulations make active security systems such as Anti-lock Breaking Systems (ABS), Airbags, and Traction Control Systems (TCS) mandatory.

Traffic management also brings security. 1994 estimated US traffic jam costs were \$100 billion and accident

costs were another \$70 billion⁽³⁾. New Intelligent Transport Systems (ITS) that help drivers with directions and traffic updates will make driving easier and will also save costs.

End-user Requirements

Apart from regulations, drivers are requiring more comfortable and easy-to-use cars. They will avoid traffic jams and will easily travel using GPS navigation systems. Active Noise Reduction (ANR) systems will provide comfortable travelling conditions. This system uses a microphone, speakers and a microprocessor to generate phase opposition signals in order to remove any disturbing noise.

Competitive Positioning

Electronic technologies will bring economy of scale and differentiation to car manufacturers. Fewer components will be used for an even wider range of systems resulting in an economy of scale. Anti-lock Breaking Systems (ABS) and Traction Control Systems (TCS) can be integrated to use the same sensors and data, which will lead to smaller control systems. Additionally, each car can have a customized single display panel. Replacing hydraulic circuits by electrical wire will also save production costs.

Electronics allow car manufacturers to provide more and more functionality for end-users. Systems such as intelligent gearboxes, Head-Up Displays (HUD) or intelligent lightning systems will be tomorrow's first buying criteria. And as the day approaches that these electronic systems become standards in automobiles, competition will drive prices down even more.

TOMORROW'S AUTOMOTIVE SOFTWARE

Today's automotive software systems are mainly developed independently of each other, which explains the low information exchange between control units. Their main tasks are data acquisition and system regulation.

(1) Industrie Automobile: La révolution électronique (Les Echos Etudes)

(2) Ford Motor Company

(3) Industrie Automobile: La révolution électronique (Les Echos Etudes)

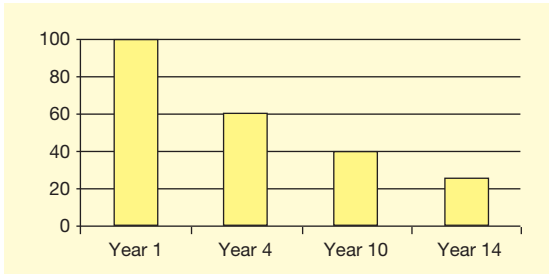


Figure 1. Automotive electronic average price (same functionality, index 100 year 1. Source EIU)

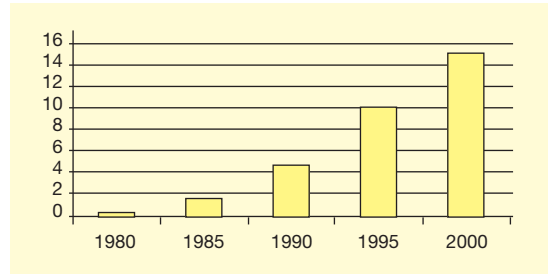


Figure 2. Ford EMS software size (in MioC: Ford Motor Company)

Tomorrow's car will be a complex embedded electronic system with an embedded network interconnecting dozens of Electronic Control Units (ECU) and communicating with the outside world via GPS-based and wireless technologies, thus bringing software complexity to unknown levels.

Embedded software becomes more and more crucial thanks to the versatility provided by microcontrollers: the same processing unit can be used for a wide range of functionality and systems. For instance, today's mechanical windshield wipers are different from one car to the next. Electronic wiping systems use the same electrical engine and angle sensors for all cars and can be configured on the production line. Car manufacturers will be able to use fewer components to perform the same function.

Today, software represents almost 50% of the overall R&D system cost and this share will increase in the future, along with software complexity, while the hardware price decreases (see Figure 1 and Figure 2).

Intelligent Transport System

Tomorrow's car will be a node in a wider information network. On-board Internet, Video-on-demand, GPS-based navigation, Automatic Toll Paying systems – all will require high volume information exchange using telecom, internet and digital imaging technologies. Most of these technologies will come from the consumer market using today's Real Time Operating Systems (RTOS) and development environments.

Electronic Control Unit Network

Future systems will include dozens of ECUs. Because of this, data exchange and processing synchronization becomes crucial, while safety concerns remain a top priority. The Intelligent Cruise Control system cannot shut down the ABS system!

OSEK/VDX: the automotive standard

In 1993, as software started to become more and more important in automotive electronic systems, European manufacturers started to develop a new automotive software execution environment: OSEK/VDX. This open architecture covers communication (OSEK COM: data exchange within and between control units), network management (OSEK NM: configuration determination and monitoring), the operating system (OSEK OS: real-time executive for the ECUs software). Supervised by a steering committee, the OSEK initiative addresses two major automotive industry requirements:

1. portability of applications to ECUs coming from different suppliers
2. interoperability of interconnected ECUs in a distributed automotive system

A project was founded to provide the relevant test methods and tools to assess the conformance of OSEK/VDX implementations: MODISTARC (Methods and tools for the validation of OSEK/VDX – based DISTRIBUTED ARCHitectures). More information on OSEK/VDX and MODISTARC can be found at www.osek-vdx.org

Most of the European car and automotive equipment manufacturers have adopted the OSEK standard for their next car's generation. Companies such as Daimler-Chrysler, Siemens, OPEL and BMW are the early adopters, but the French automotive industry, with PSA, Renault, and VALEO, is also heavily involved.

Two Paradigms

Next generation automotive software will mix event-driven behavior (the control unit reacts when receiving an event) and time-triggered behavior (how the control unit acts on a regular basis).

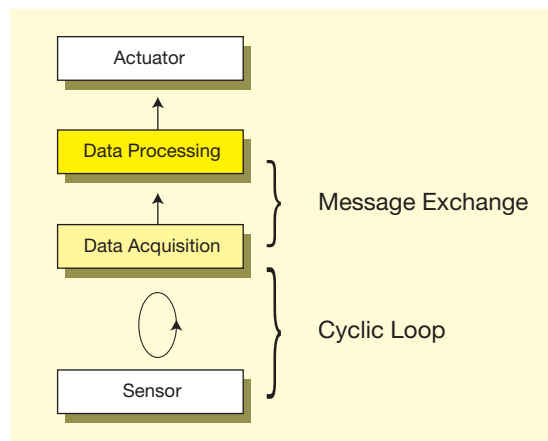


Figure 3. ECUs Event-driven and Time-triggered behaviors

Most systems use a time-triggered paradigm for sensor reading (engine control, ABS) and event-driven behavior for inter-unit exchanges. Sensor values are captured at each given timeframe and data processing occurs when the ECU receives the proper message. An ideal development environment should be able to mix both software behaviors.

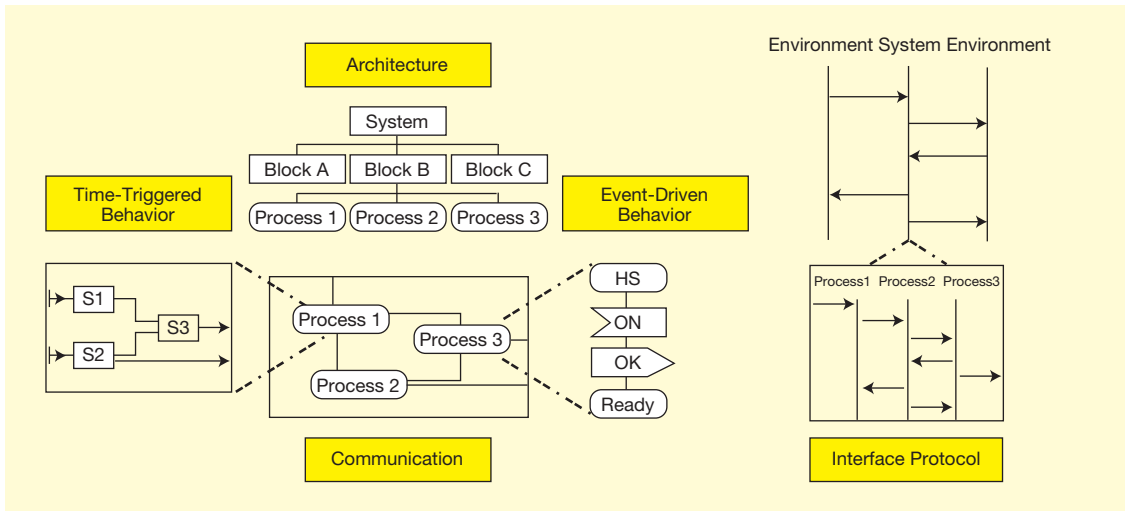


Figure 4. Notations combination

AUTOMOTIVE MARKETS NEEDS

Developing this new generation of embedded software requires larger teams and new skills. It is a very complex task because of the size, the numerous interactions between the ECUs and the external wireless technologies, and the mix of event-driven and time-triggered software.

Software Efficiency and Reliability

Automotive software has to require as little hardware resource as possible and also provide functionality. Unlike computers, cars cannot be reinitialized every 10 miles or have a crash due to a powertrain component bug. No manufacturer can afford to recall thousands of cars or market share loss. Tomorrow's electronic components must reach the same quality levels as today's mechanical systems even though they will be far more complex.

One major challenge will be to prove software safety and reliability. Certification bodies test today's cars against safety regulations: they mainly measure mechanical resistance in crash tests. But it is virtually impossible to exhaustively check software due to the millions of possible states of the systems. The automotive industry will surely move towards a certification process such as the avionics' DO-178B software development process requirements regarding safety and reliability.

Information Flow

Today's development projects are so highly complex that they are shared among numerous teams, making the information flow critical. But, textual requirements cannot claim to be correct and exhaustive. So, part of the solution comes from good notation; it has to be graphical, intuitive, and executable to ease frequent model exchange between teams or equipment manufacturers.

Another important information flow is the non-model based source code. How can we check a subcontractor's software module against specifications? How can we ensure that the proper development process was used? In other words, do we only look at the source code, or do we also need to know how it was developed?

Standard notations

Since the automotive industry needs to exchange models, they have to be standardized or make the whole industry adopt the same proprietary notations. Although StateCharts have been used to some extent, tools based on such notations cannot address all requirements, especially for communicating systems. Only a combination of notations will model all aspects of a system: components, architecture, communications, event-driven and time-triggered behavior, and interfaces (see Figure 4).

SOFTWARE-BASED SYSTEM DEVELOPMENT PROCESS

Tomorrow's automotive software will be highly complex, implying numerous interactions. This requires new software development technologies and processes. Highly complex systems development problems appear mainly at the integration phase. Systems validation and integration testing takes a large amount of time and money. An ideal development process would be model centric. This facilitates specification validation early in the development process. Correct and exhaustive specifications are the cornerstone for parallel development removing most of the integration surprises!

LEVERAGE ON EXISTING TECHNOLOGIES

The automotive industry can rely upon telecom and avionics software development technologies:

1. Car and airplane ABS share many characteristics and constraints. In the Avionics industry, software safety and function availability are mandatory. To achieve determinism, most systems are time-triggered.
2. An automotive body controller shares many characteristics with a telecom switch: information flow management between entities. Most Telecom systems are distributed over a network with highly complex systems' interactions.

Some well supported international standards provide intuitive and easy-to-learn graphical notations:

1. Unified Modeling Language (UML) from the Object Management Group (OMG) is a worldwide de-facto standard allowing graphical views of system's entities.
2. Specification and Description Language (SDL) from the International Telecom Union (ITU) is a visual formal language that has been used for years in the telecom industry. This language is particularly suited for real-time, event-driven systems, and was used for the OSEK Network Management (OSEK NM) and OSEK Communication (OSEK COM) layers definitions.
3. Message Sequence Charts (MSC) from the ITU is a natural graphical notation describing interactions between entities.
4. The SCADE notation is a graphical formalism that provides a hierarchical block-diagram approach to describe time-triggered and data-flow based systems.

SOFTWARE DEVELOPMENT SOLUTIONS

Notations need tool support when it comes to real-life projects. Founded in 1984, CS VERILOG is focused on Telecom and Avionics software development solutions.

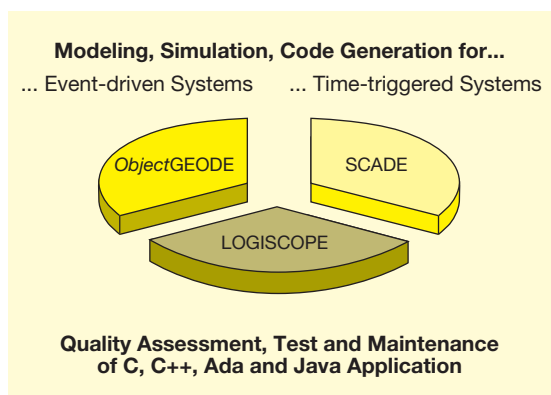


Figure 5. CS VERILOG's Tools

ObjectGEODE

Already heavily used in the Telecom market, *ObjectGEODE* is perfect for the automotive industry as well. It is a visual development environment for analysis, design, model checking, code and test generation of distributed real-time systems. *ObjectGEODE* supports a coherent integration of complementary object-oriented and real-time approaches based on the UML, SDL, and MSC worldwide standards. These notations offer a hierarchical and graphical view of all the aspects of a real-time distributed system (entities, architecture, communications, behavior and interface protocol).

ObjectGEODE helps engineers get the design right the first time through rapid prototyping, verification, and validation techniques. Rapid prototyping verifies that the system works as expected in a limited number of nominal cases. The aim of verification is to determine whether the SDL model will run reliably, regardless of

the service it is supposed to perform. Arithmetic errors (a pointer out of bounds or a division by zero), dead source code, queue overflow, deadlocks, and livelocks can be automatically found. Validation proves that the SDL description fulfills all of the system specification requirements.

ObjectGEODE automatically generates distributed OSEK-compliant code using the OSEK COM protocol over CAN or VAN networks. The middleware is delivered royalty free and can be easily customized for special needs. *ObjectGEODE*'s powerful code generator allows software architecture exploration: from the same design it can target a single or multi ECU system at a click of a button. The generated code is readable and fully executable.

SCADE

The SCADE (Safety Critical Application Development Environment) toolset comes from years of expertise in the Avionics market. It is used for time-triggered, safety-critical applications, such as autopilot or engine management systems. SCADE is based on a formal technology developed in cooperation with Airbus and Schneider Electric, and has been used to safely design many avionics (Eurocoptère, Airbus), rail transport (Ansaldo, CSEE Transport), nuclear power plant (Schneider Electric) and elevator (OTIS) systems. This track record proves that SCADE is a proven solution.

The SCADE simulator helps to detect design errors early in the development process and test efficiency is measured using test coverage. The SCADE code Generator translates the block-diagram design into very readable, reliable, and safe C or Ada code: deterministic behavior is guaranteed. The generated code is fully commented and can be easily traced back to specifications. The C code generator is the only DO-178B Level A certifiable (avionics most stringent safety regulation) code generator on the market.

LOGISCOPE

LOGISCOPE is a set of tools that provide quality and test analyses through software metrics, test coverage, and graphical reverse engineering, to facilitate competent software production and maintenance.

LOGISCOPE *RuleChecker* allows developers to automatically check their code against a set of project-defined programming rules such as the MISRA guidelines or safety-critical coding rules used in the Avionics industry. This technique guards against language traps and code misunderstandings.

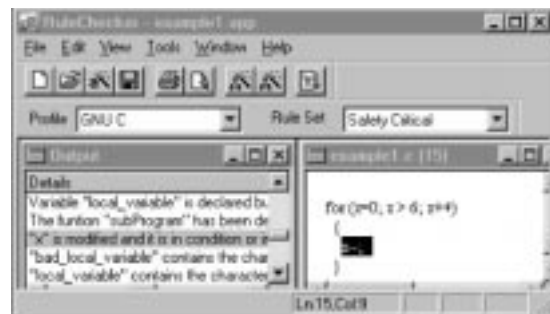


Figure 6. LOGISCOPE RuleChecker

The *Audit* module locates error-prone modules. Software metrics and graphs help diagnose defects and aid in the decision to rewrite or to deeply test the identified module. Customizable software metrics templates called "Quality Models" are also used to evaluate a program's quality characteristics such as maintainability, readability, and testability.

LOGISCOPE *TestChecker* significantly improves code reliability by identifying untested parts of code. For better test productivity, it identifies inefficient test cases and the tests that should be re-executed for regression testing if a function or a file is modified. It implements Modified Conditions/Decisions Coverage (MC/DC) as recommended by the DO-178B Level A Directives⁽⁴⁾.

Tools interaction

Existing code modules (already developed source code such as sensors' data acquisition drivers) can be directly called in an *ObjectGEODE* design and can communicate with SCADE and *ObjectGEODE* generated tasks through the operating system mechanism (i.e. OSEK COM protocol). *ObjectGEODE* tasks are the event-driven parts of the system. Tasks' and ECUs' communication mechanisms are automatically generated from the design. Those mechanisms are implemented in a Real Time Library (RTL) delivered in the source code, and can be customized to fit specific needs such as CAN or VAN networks. SCADE tasks are the time-triggered parts of the system. They can exchange messages with any other task through the RTOS communication mechanisms.

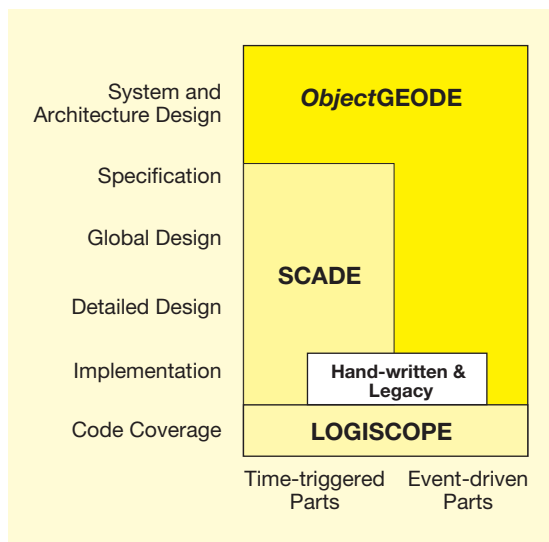


Figure 7. CS VERILOG's tools interactions

(4) A specific source code coverage method recommended by the RTCA for software certification in airborne systems and equipment. Avionics software levels of severity for potential failure distribution range from Level A to Level E, with Level A – errors that cause a catastrophic failure condition – being the most critical.

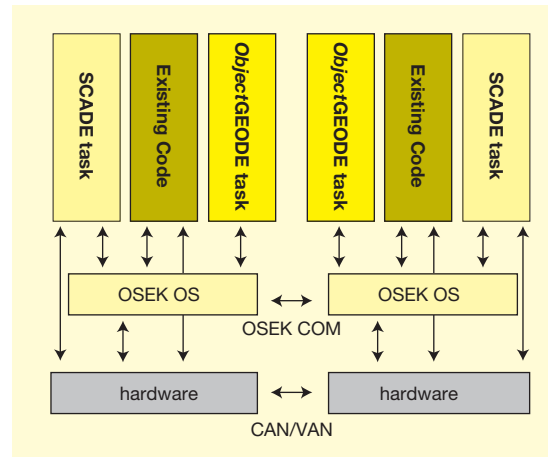


Figure 8. Code interactions

For example, if the existing code modules were sensor-reading drivers, SCADE would be used for data acquisition functions, and *ObjectGEODE* for the data processing functions and actuator activation. A SCADE task would use existing code to get the input, and alert the *ObjectGEODE* task when the environment state changes (i.e. more rain, please run the wiper system faster).

CONCLUSION

CS VERILOG is the worldwide leader in visual development environments for Avionics software and has a leading position in the telecom market. Our long-term commitment to the automotive industry is based on our existing tools and expertise together with a strong partnering network. SYSGO GmbH focuses on automotive software since years and now market an OSEK compliant RTOS, CS SI and DDS offer a wide range of development services and all OSEK major RTOS vendors are already partnering with CS VERILOG on other markets (WindRiver, ISI and Accelerated Technology Inc.). ■

Erwan Paccard holds a degree in Software Engineering from INSA (French National Institute of Applied Science) and a Masters Degree in High-Tech Marketing from EM Lyon (www.em-lyon.com). He is now in charge of the Automotive Market with CS VERILOG.