EXPERIENCES FROM MODEL BASED DEVELOPMENT OF DRIVE-BY-WIRE CONTROL SYSTEMS

Per Johannessen[1], Fredrik Törner[1] and Jan Torin[2]
[1]*Volvo Car Corporation, Department 94221, ELIN, SE-405 31 Göteborg, SWEDEN;*
[2]*Chalmers University of Technology, Computer Engineering, SE-412 96 Göteborg, SWEDEN*

Abstract: Research in distributed dependable control systems within the automotive industry is of high importance today. One reason is the introduction of more mechatronical systems. Volvo Car Corporation and the Royal Institute of Technology initiated a joint project in October 2002 to target this technology change. The project was named FAR, which stands for Function and ARchitecture integration. FAR focused on the development of drive-by-wire systems using model based development. The deliveries from the project were a tool chain for automatic code generation from Matlab Simulink and Matlab Stateflow models and also a prototype vehicle in scale 1:5. It was a very successful project and the result was delivered to Volvo Cars in June 2003. The project deliveries have been further developed at Volvo Cars since then. Primarily, a new hazard analysis method has been developed and new fault tolerance mechanisms have been implemented.

Key words: dependable systems; drive-by-wire; model based development; hazard analysis; redundancy; fault tolerance; electrical architecture; time-triggered CAN; case study.

1. INTRODUCTION

The automotive industry faces new challenges as more functionality is implemented using mechatronical solutions. At the same time, challenges as increased complexity and high dependability requirements must be handled. The dependability requirements will be in the same order as for fly-by-wire systems. It is also crucial to meet low development costs, short development time, high degree of reusability, and quality targets.

Therefore, the automotive industry needs new approaches in product development and engineers skilled to develop mechatronical system. These were the main reasons for initiating this project. In the FAR project, Volvo Cars and the Royal Institute of Technology reached these objectives in a successful cooperation.

There were many lessons learned and valuable outcomes from the FAR project. The model based development used, allowed the project to handle the complexity of the given task. The prototype car that was developed has been used to implement and evaluate several design concepts like a fault tolerant electrical architecture and redundancy strategies. Further, the car implemented the time triggered CAN protocol, TTCAN. TTCAN gave the required support to synchronize the nodes in the cluster.

When the first phase of the FAR project was finished, the prototype moved to Volvo Cars in Gothenburg. At Volvo Cars, the main research was in the area of electrical architectures. It is mainly this work that is presented in this paper. Particularly the focus has been on an actuator based hazard analysis and fault tolerance mechanisms that uses inherent redundancy. This hazard analysis was used to guide the design of the implementation.

The paper starts with a short introduction to the dependability approach including hazard analysis and fault tolerance mechanisms in Section 2. Section 3 describes different architecture views that were developed and implemented in the prototype. The prototype is described in Section 4 and Section 5 summarizes the conclusions from the project.

2. DEPENDABILITY APPROACH

The development of safety critical mechatronical products require structured design methods to assure system dependability. In this work, the focus was on an actuator based hazard analysis and specific redundancy strategies for fault tolerance.

2.1 Actuator Based Hazard Analysis

In the early stages in the design process, an actuator based hazard analysis was performed. The method used has been developed from the work by Johannessen (2001) and Papadopoulos (1999). Since it is the actuators that affect the system's environment, this actuator based approach is the logical approach for an early hazard analysis.

The failure classes used in this analysis are; *omission, commission,* and *stuck.* These classes are the worst case failures for any mechanical actuator. The class *omission* is interpreted as no energy is available at the actuator, *commission* is interpreted as maximum energy is applied to the actuator, and *stuck* is interpreted as a mechanical locking.

The chosen failure classes represent the three main failures of an actuator. Therefore, the analysis can indicate which state is the preferred fail state. For instance, if a brake failing in an *omission* state is less severe than both the *commission* and *stuck* states, then *omission* is the preferred fail state for the brake actuator. All failure classes are applied to each actuator and the system effect is analyzed.

The used severity levels are described in IEC-61508 (IEC 1998). They are *Catastrophic, Critical, Marginal,* and *Negligible.* These failure classes are used in a unique criticality ranking where the distribution between the severity levels for each failure class is considered. The sum of the distribution terms is 100%. This can be seen in the example in Table 1.

To be able to do a quantitative analysis, each severity level is assigned a weight, as shown in parenthesis in Table 1. The weights are application dependent, e.g. *Negligible* is of higher importance in a consumer product than for an industry product. The product of the severity level and the distribution numbers are added to a criticality number for each failure class.

Table 1. Example of actuator based hazard analysis for one brake actuator.

Failure Class	System severity	Distribution	Criticality	...
Omission	Catastrophic (10)	70%		
	Critical (6)	20%	8.4	
	Marginal (2)	10%		
	Negligible (1)	0%		
Commission	Catastrophic (10)	90%		
...			...	
Stuck	Catastrophic (10)	80%		
...			...	

The hazards that have a criticality that exceeds a predetermined threshold need to be handled. This method also supports a solvability analysis, where different design solutions are compared with each other. The analysis gives an indication of the best soultion.

In the FAR project, this hazard analysis gave valuable input to the design and implementation, particularly for fault handling concepts at the actuator level.

2.2 Redundancy Strategies for Fault Tolerance

The redundancy strategies used are described by Johannessen (2003) and Forsberg (2003) and include inherent redundancy, scalable software redundancy and local redundancy. These redundancy strategies are implemented in a top down approach starting with the most cost efficient strategy, which is the inherent redundancy. The most expensive approach, local redundancy, is used to fulfill fault tolerance requirements of permanent faults when inherent redundancy is impossible.

The inherent redundancy requires fail-silent actuators to be efficient. In FAR this is achieved by using wheel nodes that monitor each other by a fault handler module. The front and rear wheel nodes are grouped together to achieve a more dynamically stable system. The disable signals are directly connected to the actuators and are activated by the monitoring node to avoid unintended behavior. It is vital that the signal is an active action by the monitoring node. This approach requires a sane node to shut down the controlled node. A schematic of this solution is shown in Figure 2. An alternative solution that was implemented was to reset the controlling node instead of disabling the actuator. However, this approach was analyzed to be less safe since a reseted node has to reintegrate in the system. Therefore, it was used as a secondary fault handling mechanism.

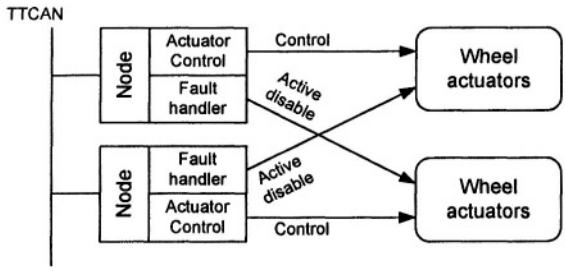

Figure 1. The monitoring node concept used to achieve fail-silent actuators.

3. ARCHITECTURE VIEWS

The FAR architecture is a further development of the Sirius 2001 architecture (Johannessen 2003) and the conceptual study of a distributed JAS 39 Gripen architecture (Forsberg 2003). This updated FAR architecture contains several different views. In this section, the functional, logical, hardware, software, deployment, and TTCAN views are described.

To capture the requirements of the system, a functional view is first developed. The logical view is highly integrated with the vehicles dynamic functionality developed in Matlab Simulink and Matlab Stateflow. One further step towards the implementation is the software view that integrates all software components in the complete system. The hardware view describes the target system onto which the functions developed in the Matlab tools will execute. To integrate the functional, software and hardware systems, a deployment view is needed. This view is also important when implementing fault tolerance and redundancy. The TTCAN view describes the communication system used in the project.

3.1 Functional View

UML Use cases were used in the early design phases to capture the requirements of the project. Figure 1 shows the project's Use case diagram. This diagram includes three users; the *Project Stakeholder,* the *Driver* of the car, and the *System Developer.* The *Project Stakeholder* is interested in the project as a whole, for visualizing new technology, education and research activities and also for marketing purposes. Both the *Project Stakeholder* and the *System Developer* can be a *Driver,* who operates the prototype vehicle. The *System Developer* is the engineer that develops the system and uses it for experiments.

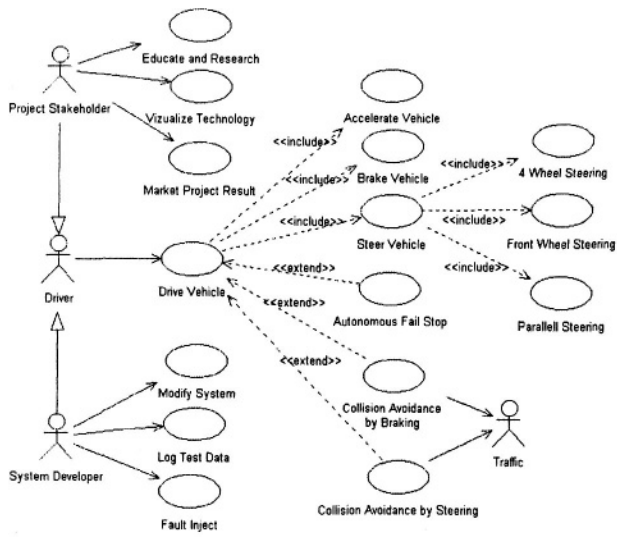

Figure 2. The FAR UML Use case diagram

3.2 Logical View

The logical view shown in Figure 3 is the basis in the scalable software redundancy strategy described by Johannessen (2003). Further, many control-by-wire systems can be modeled and designed according to the model in Figure 3. The global control functionality considers vehicle dynamics and the local control functionality handles loop closure for all actuators. All objects should be designed with as few dependencies as possible to support reusability and reduce complexity.

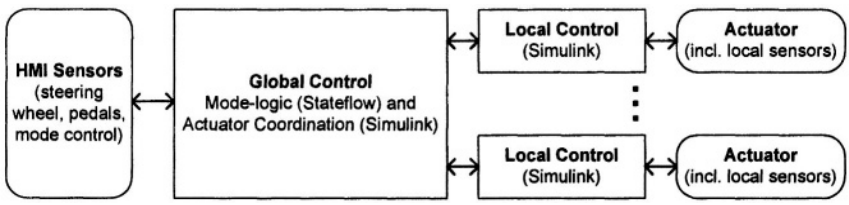

Figure 3. The logical view of the FAR architecture.

3.3 Software View

The software view in Figure 4 is the base for automatic code generation from Matlab Simulink and Matlab Stateflow using dSPACE TargetLink. Further, the clock tick from the TTCAN controller is used for distributed node synchronization. By separating the application code from the low level code such as I/O and scheduling, it was possible to automatically generate and modify application code for the target hardware.

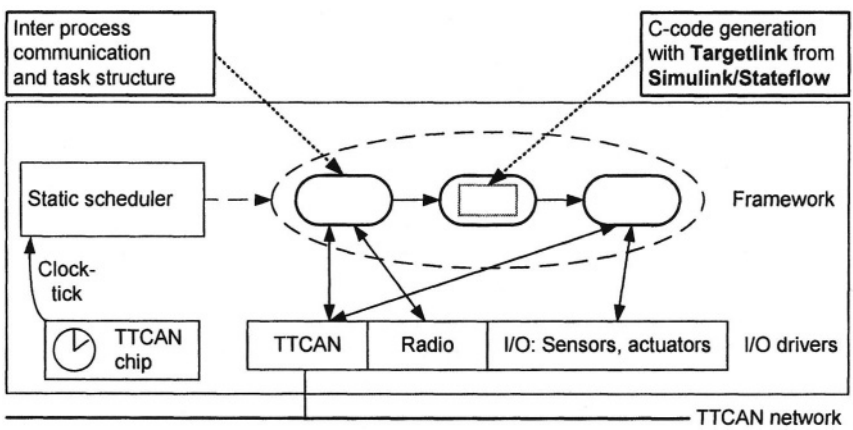

Figure 4. The software view of the FAR project.

3.4 Hardware View

The hardware view in Figure 5 shows the hardware components used in the system. There are six Motorola 68340 microcontrollers connected with a TTCAN network. The microcontrollers run at 25 MHz and are equipped with external A/D and D/A converters.

Each wheel has a dedicated node and there is one node for environment sensors. To coordinate the whole system there is one driver node that is connected through a radio link to a HMI node. The HMI node is further connected to a joystick or a steering wheel and pedals.

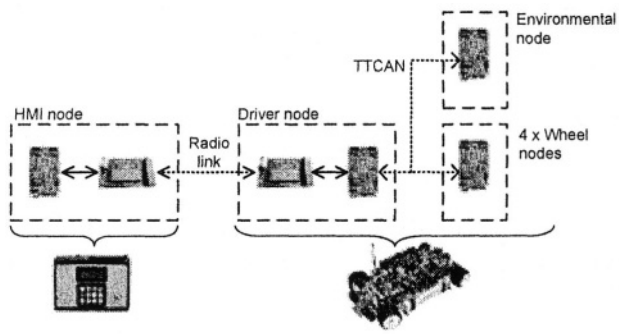

Figure 5. The Hardware view of the FAR car

3.5 Deployment View

The deployment view in Figure 6 is vital for implementing redundancy strategies and fault tolerance concepts described in section 2.2. In the scalable software redundancy concept, several instances of the global control calculations are executed in the distributed system and the results are shared in the cluster using a broadcast communication system.

Since all results from the global calculations are broadcasted, all nodes have a consistent view of the system. These broadcasted results are voted on in each wheel node, which gives a high degree of fault tolerance for transient faults.

Further, the par-wise monitoring and fault handler in Figure 2 was implemented. This applies to the brake, steer and drive actuators. Figure 6 shows these mechanisms for the front right node denoted FR. The other nodes are symmetrically identical.

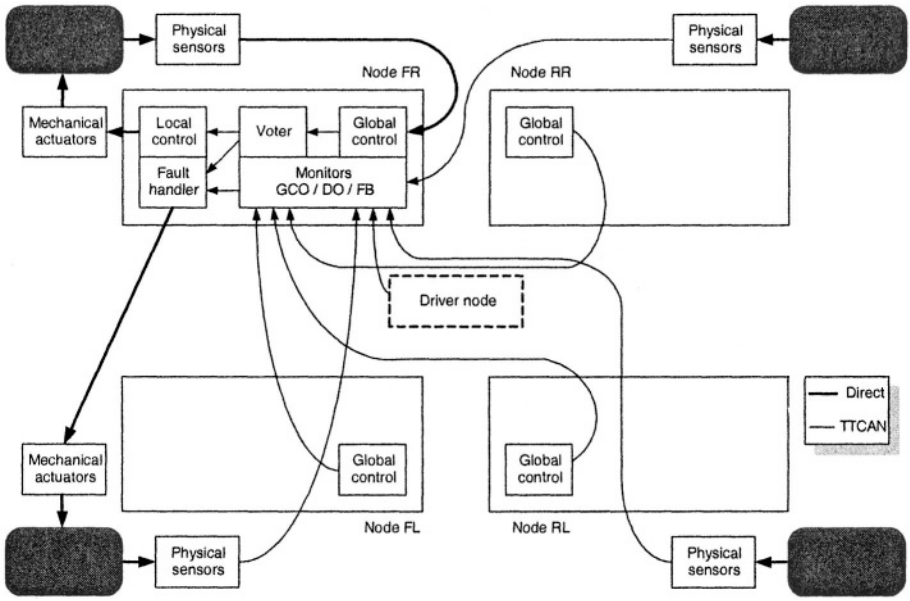

Figure 6. The deployment view of the FAR project with fault handling strategies for the front right wheel node.

3.6 TTCAN View

The communication protocol that was implemented in the FAR platform was TTCAN (ISO 2003). TTCAN was chosen since it is a potential protocol for the automotive industry that would fulfill the system's requirements. The protocol supports both time triggered and event triggered operation. Event triggered operation is implemented in the protocol using standard CAN arbitration mechanisms.

TTCAN is particularly useful in distributed control systems since the TTCAN controllers can provide a clock tick. These clock ticks can be used to synchronize the nodes in the cluster. All clocks in a TTCAN cluster are synchronized by CAN messages distributed by redundant time masters. Further, time triggered communication is predictable in the time domain.

The FAR car uses time triggered operation and approximately 20% of the available bandwidth. However, the communication could be further optimized. The cycle time of the communication system was 32 ms and consequently the global cycle frequency was 31 Hz.

4. PROTOTYPE

The developed car prototype has four-wheel steering, individual braking and four-wheel drive, with a total of three actuators per wheel. The car, shown in Figure 7 can be programmed into several modes of operation, including three different types of steering and three different types of wheel drive. This prototype will be valuable as a basis for future projects with drive-by-wire research.

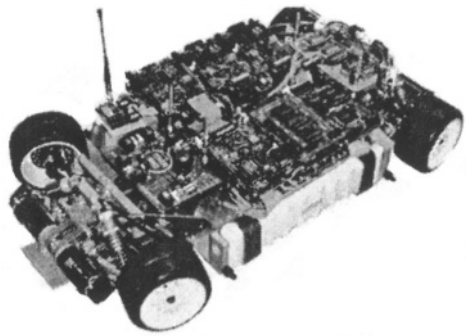

Figure 7. The developed prototype vehicle in scale 1:5.

5. CONCLUSIONS

This project allowed us to verify some concept in the development of drive-by-wire systems. Primarily, many dependability increasing concepts were validated, both to provide fault tolerance and also to be implementable in an embedded system.

The developed hazard analysis method also proved useful in the development process. It efficiently identifies real and critical failures that need to be handled. This is a requirement in the development of safety critical systems. By combining criticality and distribution of criticality for each failure class, valuable information could be obtained.

The use of TTCAN gave valuable insights. It is a highly interesting protocol, not only as a replacement of traditional CAN, but also as a communication protocol for safety critical real-time systems.

The developed prototype vehicle has shown to be as useful as expected. It is always preferred to have a real system when demonstrating new functionality or implementing new concepts to increase understandability.

It is also important to consider the teams developing the drive-by-wire systems. These mechatronical systems have a very high degree of complexity and also many degrees of freedom that give a larger possible solution space. The designers need some form of tools to handle the complexity. In this project the complexity was handled using several architectural views. Since each view considered only one aspect the complexity was manageable by the designers.

ACKNOWLEDGEMENTS

We wish to thank Professor Törngren and his students at the Royal Institute of Technology who built the FAR car and developed the framework for the automatic code generation. Further, we wish to thank Roger Johansson at Chalmers University of Technology for his support on the hardware used.

The project was funded by The Program Board for the Swedish Automotive Research Program at VINNOVA, the Swedish Agency for Innovation Systems.

REFERENCES

Forsberg, K., *Design Principles of Fly-By-Wire Architectures,* PhD. Thesis, Dept. of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, 2003.

IEC, *IEC-61508: Functional safety of electrical/electronic/programmable electronic safety-related Systems,* International Electro-technical Commission, 1998.

ISO, *Working Draft ISO 11898-4 - Road Vehicles – Controller Area Network (CAN) – Part 4: Time Triggered Communication,* International Organization for Standardization, 2003.

Johannessen, P., System Safety Design of the SIRIUS 2001 Drive-by-Wire Car – In Retrospect, *Proceedings of the International System Safety Conference 2003,* Ottawa, Canada, 2003.

Johannessen, P., Grante, C., Alminger, A., Eklund, U., and J. Torin, Hazard Analysis in Object Oriented Design of Dependable Systems, *Proceedings of the 2001 International Conference on Dependable Systems and Networks,* Goteborg, 2001, pp. 507-512.

Papadopoulos, Y. and McDermid, J., Hierarchically Performed Hazard Origin and Propagation Studies, *Proceedings of SAFECOMP'99, 18th International Conference on Computer Safety, Reliability and Security,* Toulouse, 1999.