PALBUS: An Experience Report on Designing and Analyzing Dependable Distributed Control Systems

Håkan Sivencrona, Chalmers University of Technology
Johan Hedberg, Swedish National Testing and Research Institute

Abstract

The use of distributed safety critical applications increases constantly. This new trend requires detailed knowledge about how to design the system to reach dependability. A failure in embedded distributed computer systems can result in high costs, e.g. goodwill for the company, loss of market shares, redesign of the system and most important, failure to meet personal safety criteria. Dependability can be achieved through many methods and techniques but first after a thorough requirement analysis of the system. It is furthermore important to measure and assess these properties of the system. The Swedish research project PALBUS handles these issues, how to design, develop and analyze distributed control systems with dependability in mind. The project faces dependability considerations from various aspects, for instance the importance of clear requirements and consistent terminology, wise utilization of existing protocols, e.g. CAN, intelligent fault models and development principles to reach specified fault tolerance and finally through thorough methods and techniques for verification and validation of the implementation. Knowledge from many different companies is distributed through their specific knowledge about for example communication protocols. This paper describes the PALBUS project and extracts information that could be used for dependability assessment. To do a thorough description of dependability assessment is beyond this paper whose purpose is to give a survey and some interesting suggestions concerning dependability.

1 Introduction

In distributed computer-based systems there is sometimes an assumption that these systems are safe and dependable once tested and running. This is not true since there can be no such system that would not fail under some circumstances. This assumption may however work for non-safety critical applications where it is possible to discuss in terms “loss of customers due to failures in the system”. In safety critical systems this reasoning is impossible and would not only jeopardize lives, it would also leave the company with zillions legal suits and in the end without business.
The consequences of serious failures set dependability in focus for designers of safety critical systems. This is probably the reason why more and more companies are interested in how to assess and prove dependability properties.
Still [illegible] hide their heads in the sand like some famous bird.

How to assure that the system is dependable enough?

PALBUS, a Swedish joint research project between industry, university and research institute, addresses this issue and discuss methods and principles for dependability in embedded systems such as CAN.

PALBUS investigates issues like the importance of using a common and well defined terminology relevant standards, motivations for the choice of protocol for a specific application, design principles, implementing dependability, fault handling and detection, verification and finally validation methods.

This paper is a scan over the project parts and should be useful for a designer of [illegible]

interesting questions for future related projects.
The paper first consists of terminology, comprehension and specification part and then embedded control systems to choose a protocol concept. Next step is Fault model, handling and detection followed by system design principles. After these parts comes, validation and verification of the system. Finally system development procedures and system aspects.

2 Terminology, comprehension and specification

Relevant standards and an appropriate terminology are basic for a safety critical system design. Standards because they set requirements for a specific application. The terminology in PALBUS is based on Laprie´s [Lap95].

The terminology is even more important for distributed control systems, since many different vendors design nodes that are connected to each other and using a common communication channel to exchange safety critical information.

In order to discern dependability features of a system the following attributes are commonly used:

- Availability, which means readiness for usage
- Reliability, which means continuity of service
- Safety, which means avoidance of catastrophic consequences on the environment
- Security, which means prevention of unauthorized access and/or handling of information
- Maintainability, restoration after failures

It is very important to understand how the system will be used and which safety critical situations it must cope with and of course make sure that this knowledge is correctly implemented in the specification. To be able to reach these demands it might be necessary to introduce algorithms and techniques, such as atomic broadcast, membership agreement, fail silent nodes, redundancy (TMR) and are thoroughly explained in PALBUS [Edl00].

Following questions must be considered in the specification:

- Is it clearly defined what a specific level of dependability means.

- What actions will be taken to reach that certain level of dependability? In effect how can it be proved?

In PALBUS the importance of system comprehension is discussed and how it could be improved. PALBUS emphasis on ways to describe and comprehend functionality and dependability requirements for a distributed system. This can be done with a set of visual notations and tools [Wan00].
Comprehension is a cognitive process to really understand the requirements, the design and implementation of a safety critical applications and their behavior. Comprehension usually involves cognitive activities for understanding system notation, architecture, components and their static and dynamic behavior.

3 Embedded control systems, to choose a protocol concept

As more and more systems are designed using connected distributed computers instead of one master node, to handle and pass all messages between the distributed nodes, the need for in depth understanding of functionality, failure behavior and fault tolerance has increased. The topology of system architecture and also the communication paradigm affects the use and need of a specific design. PALBUS examines different protocol concepts and describes advantages and disadvantages with different solutions for instance, event triggered communication v/s time triggered as well as single-master control v/s multi-master system control, i.e. a fully distributed embedded system.

Whom in charge?	
Multi Master	**Central Master**
All nodes can control	One node can control
Robust	Single point failure
Low overhead?	Easy to analyze
When to access?	
Event triggered	**Time triggered**
Control triggered by events	Control in time-slots, easy to make fail-silent
Efficient for discrete events	Efficient for continuous signals
Not predictable, not deterministic	Predictable, replica determinism
Dynamic scheduling	Static scheduling, before run-time
Flexible, easy to add tasks	Composability, no side effects, rigid
Synchronized via messages	Inherently synchronized, periodically
Low average delays, limited predictability	Known delays, small jitters
Time adequate	Resource adequate
	Easy to test

Figure 1. Comparison between different architectures.

Developers of the system could use the PALBUS comparison to increase the understanding of the differences between architectures and find out which solutions are most appropriate in each application.

The topology of the communication system is basic for a dependable system structure and also affects the fault model. Some topologies, offer possibility to use double, redundant channels and are often utilized in protocols that should provide fault tolerance against bus failures, i.e. short circuits and similar. This topology need not be double throughout the whole system but between those nodes that need redundancy. If the double channels are combined with redundant nodes, it is possible to increase the reliability of the system. A uni-processor system may use a star coupled topology, and for example disconnect short-circuited wires and thus it is a robust system from that point of view. Other topologies could be uni-directional ring structures, tree structures and fully or partially connected structures.

For a safety critical application such as a brake-by-wire system, all actuators need some intelligence (processor) and neither of these nodes should be given a chance to become a single point of failure. Thus one could reason that a single master system would not be appropriate. And introducing a distributed system in this the system and require for example membership and atomic broadcast protocols to be implemented. This of course affects the dependability itself and must be handled by the fault model.
A small comparison between different communication principles (protocols) is presented in figure 2.

	Single-master	**Multi-master**
Event-triggered	Soft real-time Not deterministic None stop systems High bandwidth demands ***Mil-Std-1553B***	Safety Critical? Soft real-time Not deterministic None stop systems Low bandwidth for discrete messages ***CAN***
Time-triggered	Safety critical Fail silence Hard real-time Predictable High bandwidth demands Easy to analyze ***OBDH***	Safety critical Fail silence Hard real-time Predictable Low bandwidth for continuous signals Cost effective ***TTP/C***

Figure 2. Features of four different protocols used in safety critical applications.

In PALBUS these protocols (figure 2) where assumed to be able to handle a safety critical application, although with different efforts in h/w and s/w [Siv00]. In effect, some protocols must use resources in higher layers to reach these criteria while others may have these dependability mechanisms in the communication controller.
Analyze is needed to discover if the performance of the protocol is good enough to handle short deadlines, high latencies, jitter etc.

4 Fault model, handling and detection

A distributed system is believed to achieve better fault tolerant performance than a centralized system because the distribution of intelligence and architecture makes it easier to avoid single points of failures. A side effect does appear though. The error detection, error avoidance and error handling must be distributed as well.

During design it is important to create a fault model and possible fault states and the next step is to check that necessary error detection and error handling are

However there are often difficulties to foresee all faults and/or inject these artificially.

PALBUS presents different faults that can occur in the system and the effect of these that must be taken into account when choosing a fault model. It describes a number of different fault detection and fault handling mechanisms [Ask00].

Typical faults are temperature changes, mechanical vibrations, ageing, and electromagnetic interference that may result in transient or permanent component failure.
Design and specification faults are also a source for failures but are harder to detect. Specification faults are misunderstandings of the demands on the system and cannot be discovered by the system, since it follow the specification.
Design faults can be either hardware or software faults and occur when the design does not match the specification. To detect these faults exhaustive testing is required although one can seldom detect all. A fault tolerant approach must then be chosen implying the use of design diversity. This means that different software (N-version programming) and hardware are used for nodes with redundant functionality.
The method chosen often depends on the demands on the system. While recovery is possible for some system it may not be good enough for real time systems with short deadlines. Although some fast systems may use some sort of rollback.

The system performance when faults occur can be described from following viewpoints:

- The system must tolerate and still provide full service for a specified number of arbitrary faults. This could be a steer-by-wire system for example.

- The system should, for a specified number of faults not be degraded more than to a certain level and still maintain some service, the quality of service.

- The system should at least membership agreement) faults that may lead to failures and shut failed blocks/nodes off (could be use of non-specified plug and play devices).

- Assume that some faults are transient and continue with the function as long as extrapolation from old values can be done satisfyingly. If not the system will be turned off.

- The system shut down, if possible; i.e. there exist a fail-safe state.

Depending on the application and the criticality of the task, it should be able to combine these above-mentioned scenarios. For some systems failing nodes are assumed to become fail-silent. This is an issue for deciding about used protocol.

Ways to act after faults could be:

- Graceful degradation

- Use shadowing or redundant nodes/circuits.

- Reset of the system or recovering failing circuits/nodes.

5 System design principles

When a protocol has been chosen and the fault model with its mechanisms has been decided it is time to implement the design outgoing from a certain high level protocol. PALBUS examines actions that must be taken care of during design phase.

The thought in PALBUS is that outgoing from a certain high level protocol chosen, what actions can I as designer/developer perform to create as dependable system as possible.

The examination takes into consideration how to build up the bus topology (for instance does the system need a duplicated communication channel?). The importance of utilizing the CAN protocol optimally (Is all available status and control signals utilized which can effect dependability?). Choosing a relevant high-level protocol which can cope with the

dependability in the application code and finally make sure that the system aspects have been solved.

6 Validation and verification of the system

One of the most important parts in the development is the verification and validation of the distributed control system.

The verification should show that the system does agree with the specification and the validation shows that the system can cope with safety risks in the environment where it should be used.

Validation methods described in PALBUS could be used during validation/verification but also during development to increase dependability. Methods described are focused on validation of distributed control systems, in effect, which specific methods should be applied to distributed control systems. The system validation methods are divided in following subgroups [Hed00]:

- Formal methods
- Assessment
- Test
- Inspection & audit
- Trial
- Simulation

The purpose with this part of PALBUS is to give the industry a “tool-box” of methods, which could be used during validation/verification.

The methods described above could be used as a part of the dependability assessment. Some of the methods described are analytical and some of them imply practical testing.

Another assessment approach to use to structure the whole system and to decrease the complexity of the design, is to divide the distributed system into following levels:

- Physical installation level
- Bus level
- CAN communication controller level
- Application level
- Quality level

For each of these levels it is possible to define a number of dependability parameters which also is done in PALBUS, but it is difficult to know if all relevant parameters has been implemented on each level and depending on system different parameters could be of importance. But dividing the system into a number of levels makes the analysis easier because it is very complex to concentrate on the complete system simultaneously.

Fault injection is an example of a very useful method to observe and verify how faults can be handled by the redundancy in the system. Fault injection can be used to calibrate methods that use threshold constants for fault handling. Finally they can be used to validate the efficiency of the specified fault handling mechanisms. This is the last step.
There exist several methods to extensively fault inject and test a system, these are for example Heavy ion injection, electromagnetic injection, pin injection etc.
Fault injection is used where it is believed it can represent real faults in the sense that they can produce faults acting as those caused by real faults. The faults can furthermore more be injected at different levels in the system and thorough planning is needed. It is also a time consuming task to handle the results and interpret them to be able to modify the system to become more fault-tolerant and dependable.

7 System development procedures

How is the design selected and how are experiences handled. Who is responsible? What is logged? This is some of the fields where companies many times have developed their own methods. Companies that design real, safety critical systems must have clear and realistic requirements on both the system itself but also on their own used methods. An investigation in PALBUS shows that many companies have very different strategies of how to reach dependable systems. This investigation handles both test methods; development processes and the way

handled. As for requirement handling, this is something that can include more clear specifications.

A way to assess the development of the new system is shown in figure 3, below.

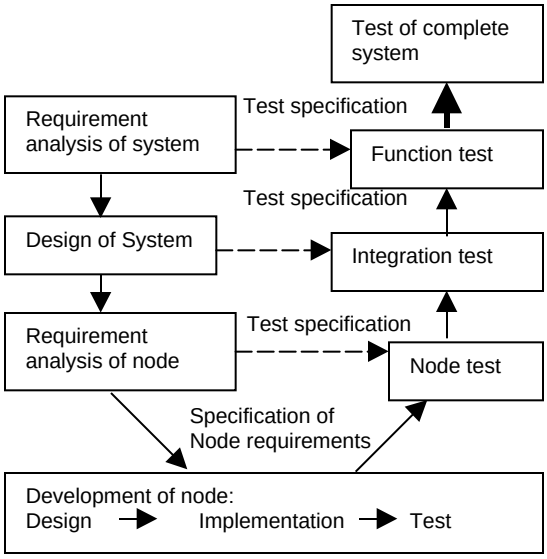

Figure 3. A process for development of safety critical systems.

In the process to assess every step in the process it is also valuable to iterate the procedure so as to get something like requirement analysis then to construction of system, back to requirement analysis and then down to requirement analysis of the node and so on.

8 System aspects

An important part of PALBUS is to consider the system aspects to reach dependability. It has arisen in the project that these issues are of great importance. Below follows a number of aspects that are considered in PALBUS:

- Could the system detect if a new node/unauthorized is placed in the cluster?
- Is it possible to check the status of each node during start up or during runtime?
- Could the system detect if a node is missing during start up or during runtime or works incorrect?
- How does the system cope with incorrect messages or disappeared
- How is the "babbling idiot" failure handled?
- Degradation of functionality during error condition, how is this solved?
- How is the synchronization between different nodes solved?
- Are errors logged somewhere in the system when they occur?
- Does the system has some kind of intrusion detection

This is just a few of those system aspects that are discussed and described how they are solved/not solved in different high level protocols based on CAN.

These system aspects are listed in PALBUS and could be used during dependability assessment [Wan00].
In PALBUS a number of important system aspects has been placed in a checklist to give suggestions

9 Conclusion

PALBUS has given a good overview of important parts for dependability consideration. It has been very developing to work together with industry, university and research institute and to recognize interesting areas. Although many institutes, universities and companies invest much effort in embedded computer system development, it is still in the beginning of the revolution. ***Dependability assessment*** is the foundation to meet liability and viability demands from both insurance companies and authorities.

The PALBUS project has furthermore given the companies a more detailed knowledge concerning how to proceed in their overall development. One aspect is the importance of a system responsible, for a single node, the interaction of several nodes or perhaps the whole system.

Bus topology, time versus/event single/multi master have massive impact on the robustness/dependability/efficiency and also that a protocol should be specified for all these combinations to give the system designer application flexibility.

There exists many development tools for

development of the systems but can the tool provider guarantee that these tools do not introduce new faults and where is the responsibility in such cases, should it be on the system developer?

Who is responsible? Are there clear instructions how the product shall be used? Is it possible to introduce the product in for example the North American market and thus face possible legal suits that may be a fact after a faulty function of the system?

There is a need to avoid ad hoc methods because these can result in the system being impossible to validate and also introduce new fault states; furthermore the dependability assessment should be done again. Plug and play components must be specified or avoided totally if you have no control over the black box.

Short time to market or a dependable system? Is this a contradiction?

How is the requirement handling done in the development process? This is something that many companies must improve. Does it exist time to iterate the process once more to spot weaknesses in the system?

What are realistic requirements? In safety critical systems a zero fault demand is unrealistic and maybe also impossible to reach. Instead focus could be set to qualitative requirements where for example no single fault may lead to accidents or quantitative requirements that could mean not more than a specified fault rate, for example.
In these realistic requirements it could also be mentioned requirements of specific development processes or claim limits for lowest unreliability.

What is required to be able to trust CAN based protocols in safety critical applications? Is it enough to add s/w in the application code or is the h/w able to meet the real safety critical demands?

10 References

[Lap95] J.C. Laprie, ***Dependability – Its Attributes, Impairments and Means:*** Springer-Verlag, 1995.

[Siv00] H. Sivencrona, J. Hedberg, ***Comparative Analysis of Dependability Properties of Communication Protocols in Distributed Control Systems:*** PALBUS10: 2, 2000.

[Chr99] M. Rimén, J. Christmansson, ***Testing CAN-based Safety-Critical Systems using Fault Injection: In Proc. Sixth Int. CAN Conference (ICC´99), section 11, pp. 16-21,*** Turin, Italy, November, 1999.

[Law92] H. W. Lawson, ***Parallel Processing in Industrial Real-Time Applications:*** Prentice Hall, 1992

[Wan00] Y. Wang, et al, ***Distributed System Dependability Description and Comprehension:*** PALBUS10:3, 2000

[Hed00] J. Hedberg, et al, ***Validation methods for distributed control systems:*** PALBUS10:10, 2000

[Edl00] H. Edler, et al, ***Definitions***: PALBUS10:1, 2000

[Ask00] Ö. Askerdal, ***Fault detection and handling:*** PALBUS10:5, 2000

Chalmers University of Technology
Department of Computer Engineering
SE 412 96 Göteborg
+46(0)31-772 1669
+46(0)31-772 3663
E-mail: sivis@ce.chalmers.se
Website: www.ce.chalmers.se

Swedish National Testing and Research Institute
Physics and Electrotechnics
Software & Safety
Box 857
SE 501 15 BORÅS
+46(0)33-165071
+46(0)33-125038
E-mail: johan.hedberg@sp.se
Website: www.sp.se/pne/software&safety