Dependability characteristics and safety criteria for an embedded distributed brake control system in railway freight trains

ROGER JOHANSSON

CHALMERS LINDHOLMEN UNIVERSITY COLLEGE
Göteborg, Sweden 2001

Report no.8, August 2001

REPORT No.8

Dependability characteristics and safety criteria for an embedded distributed brake control system in railway freight trains

ROGER JOHANSSON[1]
Department of Electrical and Computer Engineering
Chalmers Lindholmen University College
roger@chl.chalmers.se

Chalmers Lindholmen University College

Göteborg, Sweden, Aug. 2001

[1] This work has been conducted within the Centre of Excellence CHARMEC (CHAlmers Railway MEChanics) – a VINNOVA Competence Center under the "Programme area 4, SD3. Computer control of braking systems for freight trains"

Dependability characteristics and safety criteria for an embedded distributed brake control system in railway freight trains

ROGER JOHANSSON

Report – Chalmers Lindholmen University College

Report no. 8

ISSN 1404-5001

Chalmers Lindholmen University College

P.O. Box 8873

SE-402 72 Göteborg

Sweden

Telephone + 46 31 772 1000

Chalmers Lindholmen University College

Göteborg, Sweden, Aug. 2001

Abstract

This paper provides a general discussion on the subject of distributed computer control for safety critical train applications. As computer technology evolves it brings new opportunities to achieve a higher degree of efficiency and safety. One of the keys to all these benefits is a dependable computer based control system. Dependable computer systems have been essential components in; for example, aircraft control systems for many years. Thus, required techniques and methods for building such systems are well understood. The important differences in train applications are mainly economical which brings us the challenging question: how do we design and manufacture dependable train control systems in a way so that they become attractive as standard components in future railway vehicles? The objective of this paper is to investigate the fundamentals for distributed safety critical computer control in trains and establish the prerequisites for these systems.

1. INTRODUCTION

Computers have been an integrated part of train traffic systems for many years mainly in centralized control. Such systems are referred to as Automatic Train Control (ATC) systems [1]. For simplicity, ease of comprehension and the desire to maintain consistent control, coordination and control functions have been implemented into centralized systems. This concept often results in slow decision making, i.e. system output such as route information exhibits latencies that sometimes even delays the transportation units [2]. As slower decision-making implies imprecision and furthermore, centralized units are highly susceptible to natural and artificial disasters, the needs of future transportation systems must necessarily undergo radical transformations [2].

Even a single transportation unit (i.e. a train set) utilizes several different computers for automatic controls. Driver-less trains are supported by Automatic Guided Transit (AGT) as well as Automatic Train Protection (ATP) systems [3]. A single vehicle uses distributed computer systems for a number of good reasons. The main reason is that the computer adds generality in control and flexibility to most systems. Given a computer-controlled system it is beneficial to put the computing power close to the sensors and actuators that are the interface between the control and the controlled systems. Electronic systems can replace mechanical systems in a cost-efficient way and computer control can, properly handled, add safety to a system.

In the near future, we are likely to see trains where computer control largely is brought into the vehicles (locomotives and wagons) and perform safety critical functions related to the vehicle dynamics. Consequently, we will have a thoroughly computerized transportation system with all its potential benefits. There is, however, a dilemma; what if the electronic system (or part of it) fails? Clearly, the complexity and hazards of such a system have to be carefully considered.

The primary objective of this paper is to provide a general discussion on safety related computer based control (where control in all essentials applies to dynamics) in trains. A good reason for doing this is to establish an interdisciplinary base for upcoming efforts of bringing sophisticated electronic devices into new applications as well as replacing old solutions by means of new technology.

As a secondary objective, we will discuss in detail and elaborate on safety requirements placed on train brake systems. With focus on the brake systems we get this important trade off that our results become quite general since other potential computer controlled functions generally are categorized in safety critical levels below (or the same as) the brake systems.

The scope of this work is distributed embedded safety-critical control applications. In this paper, we focus on railway traffic even though most of the issues we discuss have very much in common with similar application areas.

In contexts where restrictions due to geographical or national irregularities apply, we presume European conditions, i.e. standards, recommendations and regulations from European Transport Authorities are considered.

This paper is organized as follows; in chapter two, we review important aspects and characteristics of real time systems for control applications. Essential terminology concerning the design of safety critical systems will also be recaptured. Readers familiar with real time safety critical computing might want to skip these parts and may do so without loss of continuity.

In chapter three, we discuss characteristic requirements for the train application control. The application is viewed as a distributed real time system with emphasis upon the design and development of such systems.

Chapter 4 gives a case study where we apply the general ideas discussing a brake control system.

Chapter 5 offers a discussion about available means and methods as well as an outline of our planned future work within this area.

2. COMPUTER BASED CONTROL SYSTEMS

An important field of computer exploitation is real-time systems. A real-time system can be understood as an information processing system, which has to respond to externally generated input stimuli within a finite and specified period. We say that the services the system delivers have to be valid both in the value domain (correct value) and in the time domain (result arrived in time). Expressed in other words, each computational activity in the system must meet its deadline.

A distributed real-time system can be thought of as a set of micro controllers, which communicate via a common (serial) bus. The choices of serial buses for these systems are often tradeoffs from economical and practical aspects.

An important class of real-time systems deals with control applications. The functionality of such a system may be divided into three major parts:

- Get information as soon as it is available (*input*).
- Process information, i.e. calculate new output values (*calculate*).
- Present result within the specified period, i.e. send the computed value to an actuator (*output*).

There are special time requirements emerging from typical control applications. Sensors and actuators related to the same control object may be connected to different nodes in distributed systems. Delays are introduced in the control loops, mainly because of communication delays. These might be minimised to get a high control performance. The time between sampling and actuation for a given control loop is generally required to be constant in consecutive samplings even if this means it cannot be minimal. This is because the control laws are designed with specified delay compensation. A *varying control delay* invalidates the compensation and causes reduced performance. Reduced performance and even instability is also caused by variations in the sampling frequency, *jitter*. Periodic processes for sampling and actuation must therefore be forced to execute with a fixed period. This may require a higher sampling frequency to fit in the schedule and thus a modified control law.

Various characteristics for different applications means we have to be more precise when we stipulate real time requirements. We generally say that a computation must meet its deadline but that is not enough. We speak about *hard critical* real time requirements when a missed deadline may cause a disaster for the entire system. A *hard* deadline, on the other hand, is not equally severe, but the system does not gain from a result produced after a hard deadline, i.e. the result is useless. If the result might be useful even when the program has missed its deadline, we call this a *soft* deadline. However, the system gains less and less from the result the later, it arrives.

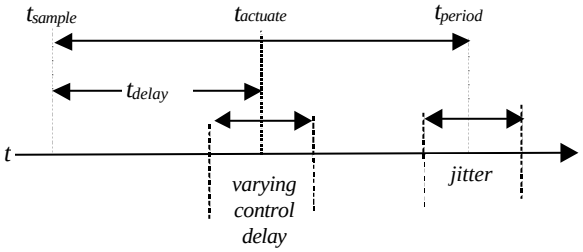

Figure 1 Varying control delay and jitter

Real time systems may be implemented according to an *event-triggered* approach, a *time triggered* approach or a combination of these approaches.

An event-triggered system is desired solely to respond as quickly as possible to the stimulus. Such systems are interrupt-driven, i.e. they have a special mechanism that gets the processor's attention by driving a signal active (interrupt-signal). This causes the processor to suspend current execution, dispatch to a predefined interrupt service routine. The service routine then performs any action required as a response to the stimuli.

Just as with an event-triggered system, a time triggered system should respond to external stimulus, a time-triggered system, however, uses a polling technique to check for the event, i.e. the system now

checks for a real-time event at regular predetermined intervals. During each interval the input device that reflects the event is monitored. Obviously, these intervals must be constructed to guarantee that all hard or hard-critical real-time requirements should be met.

By removing hardware interrupts and software interrupt handling, time triggered systems provide us with a fully time-deterministic behavior, we might exploit the system's functionality and performance at compile time. The time-triggered approach is therefore often preferred to meet hard or hard critical deadlines.

A real time system can seldom be considered either hard or soft as a whole; rather, the system delivers services, which might fall into either of the categories. This is true for most cases and especially for distributed real time systems. On the contrary, when we decide to distribute control throughout the system we often (inherently) allocate dedicated nodes to local (hard-time critical) functions. At the same time, we use the network to gather information about the entire system status and these tasks may or may not have hard time-critical requirements. These are some of the considerations that lead to the conclusion that a mix of the fundamental approaches (event triggered and time triggered) should bring the most efficient solution to a real time system for distributed control.

2.1 Dependability

Dependability is that *property of a system that justifies placing one's reliance on it* [4]. Considerations of dependability have implications for a complete system, and for all stages of its life from inception to decommissioning. Dependability is an overall property, which has other measures such as safety, reliability and availability. In [4] these terms are defined as follows:

- Safety is a measure of the continuous delivery of service free from occurrences of catastrophic failures.
- Reliability is a measure of the continuous delivery of proper service (where service is delivered according to specified conditions) or equivalently of the time to failure.
- Availability is a measure of the delivery of proper service with respect to the alternation of proper and improper service.

A *system failure* occurs when the system fails to perform its required function (continuously deliver service). System failures are caused by one or several errors'. An *error* is a deviation from the required operation of the system or subsystem. A *fault* is a defect within the system that might lead to an error. Consequently, a single fault could potentially lead to a system failure. Faults may be classified due to their nature:

Permanent the observation of a permanent fault is always reproducible and it applies to both hardware and software. The fault may have been present since the system was manufactured or emerged during operation (hardware).

Transient a transient fault could be thought of as a temporary disturbance, which is not reproducible in the general case. Transient faults apply to both hardware and software. In hardware, they can occur because of electromagnetic or mechanical shocks, while in software they can arise from certain combinations of input data for example.

Intermittent an intermittent fault is a special case of the transient fault in that it re-arrives and therefore in general becomes observable. In hardware, these faults typically emerge from slipping connectors.

When it comes to software, we sometimes speak of "Heisenbugs", as a particular nasty form of faults. Their general characteristic is that while an error is observable in a particular environment it disappears when the system is transferred to another. For example, when we introduce debug utilities, this affects the system in a way that errors emerging from the fault we are looking for become invisible.

It should be noted that a fault may or may not be observed (fault detection mechanisms are seldom perfect) while an error generally is detected. It should also be emphasized that while some system failures may have severe consequences most of them will probably not. Thus, a carefully designed

system might exhibit faults, errors and even system failures and continue to deliver acceptable services depending on its application.

There are four approaches to achieving system dependability: fault avoidance, fault tolerance, fault removal and fault forecasting, called the *means for dependability*. It is commonly agreed that a combination of these approaches must be used in order to achieve ultra-high dependability.

Fault avoidance methods apply to the construction phase of a system. These methods aim to prevent faults from occurring or even prevent the introduction of faults. Tests and validation reside within this category.

Fault tolerance methods use redundancy to maintain delivery of services even in the presence of faults. We often use terms such as Fail Operational (FO) and Fail Safe (FS) (or fail silent) to express the degree of fault tolerance. The FO property means that a system will continue to deliver uninterrupted service without loss of performance even in presence of a single fault regardless of where (in the system) the fault emerged. The FS property means that a single unit turns itself off and does not deliver services at all in presence of a single fault. This property is particularly important in distributed control systems and will be further discussed in chapter 3.

Fault removal methods are used to minimize the presence of faults using different formal verification techniques. Many of these methods apply to software. While the potential use of such methods would be of tremendous value they are still not capable of handling real systems due to their large complexity.

Fault forecasting methods use evaluation to estimate the presence, the creation and the consequences of faults. For example, statistical methods can be used to predict the probability of a future system failure based on observations of errors during earlier system operation.

When designing a dependable system in general the design process may be divided into four activities:

- abstraction, identifying the essentials
- decomposition, reducing objects into a number of simpler, smaller parts; analysis of interactions, interfaces and structures; modularisation
- elaboration, adding and detailing features
- decision making, identification and selection of alternative strategies

2.2 Safety critical systems

A particularly important class of dependable systems are *safety critical systems*. The development of a safety critical system may be considered as a series of transformations of its definition. These transformations are often described with the 'V-model' of the development lifecycle (see Figure 2 below).

As the work proceeds from the requirements through to its implementation, work is organized in phases where each phase takes as input a description of the system and develops this to form input to the next stage. In order to have confidence in the final system, the work performed during each phase must be verified.

Verification is the process of determining whether the output of a phase fulfills the requirements specified by the previous phase. The task of verification is to demonstrate that the output of a phase conforms to its input rather than to show that the output is actually correct. Consequently, errors in the original requirement will propagate without notice through verifications.

Validation is the process of confirming that the specification of a phase, or the complete system, is appropriate and is consistent with the original requirements. Thus, a validation of the complete system demonstrates its suitability for use and confirms the appropriateness of original requirements.

Verification and validation are achieved by inspections and testing. Results from these procedures may be used to investigate safety characteristics for example. Testing may also uncover faults that may then be removed. Testing is performed at various stages during the development of a system and its role

in safety assessment makes testing an overpowering effort. This effort includes *careful test planning* with detailed descriptions of test activities as well as prescribing the different test methods to be used. It is, however, important to realize that testing itself is insufficient. Tests may only be used to demonstrate the presence but not the absence of faults.

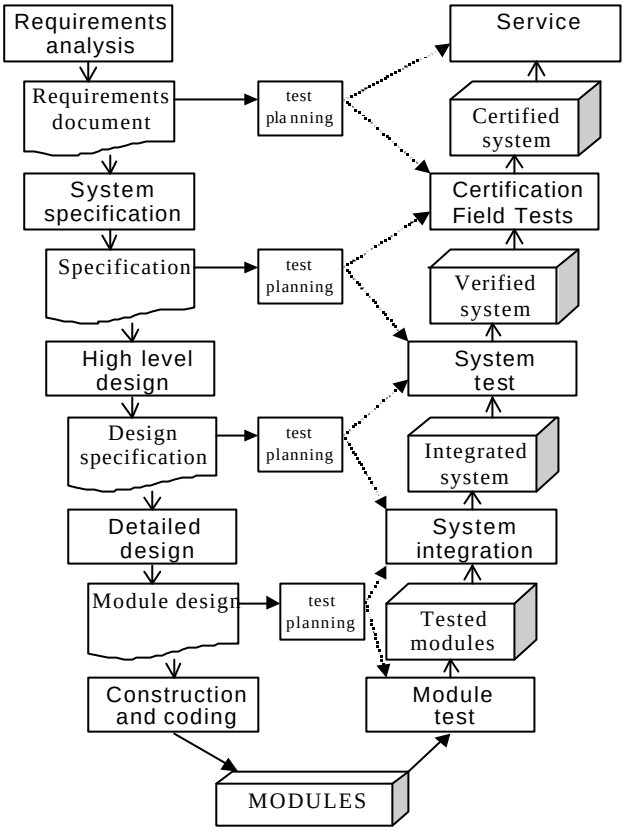

Figure 2: The 'V' development lifecycle

Dependability requirements are established during the overall requirements analysis phase. This is the phase where situations that could possibly lead to catastrophic failures should be identified.

A *hazard* is a situation in which there is an actual or potential danger to people or to the environment. In other words, a hazard might potentially lead to a possibly severe accident. Associated with each hazard is a certain risk, related to the likelihood of the event occurring and to its likely consequences. During requirements, analysis hazard analysis must be undertaken. Among the most widely used techniques are:

- Failure modes and effects analysis (FMEA), a qualitative method of reliability analysis which involves the study of the failure modes which can exist in every sub-system of the product. FMEA also tries to determine the effects of each fault mode on other sub-systems and on the required functions of the product.
- Failure modes, effects and criticality analysis (FMECA) is an extension of FMEA where consequences of particular failures are considered. This allows efforts to be directed at the areas of greatest needs.
- Hazard and operability analysis (HAZOP) investigates effects of deviations from normal operating conditions.
- Event tree analysis (ETA) tries to identify the events that can affect the system and tracks them forward to determine their possible consequences.
- Fault tree analysis (FTA) establishes a set of possible fault modes or external events that may result in a stated fault. This can be thought of as ETA in reverse direction

Other system development processes include object-oriented techniques. Such processes exhibit iterative and incremental cycles. A project is structured into time-base phases such as:

- *Inception* - specifying the project vision
- *Elaboration* - planning the necessary activities
- *Construction* - building the system as a series of incremental iterations
- *Transition* - delivering the system for services

Likewise the project is structured along the process component dimension, including the following activity:

- *Requirements capture* - a narration of what the system should do
- *Analysis and design* - a description of how the system will be realised in the implementation phase
- *Implementation* - the production of the code that will result in an executable system
- *Test* - verification of the system

System development proceeds as a series of iterations that evolve to the final system (Figure 3).

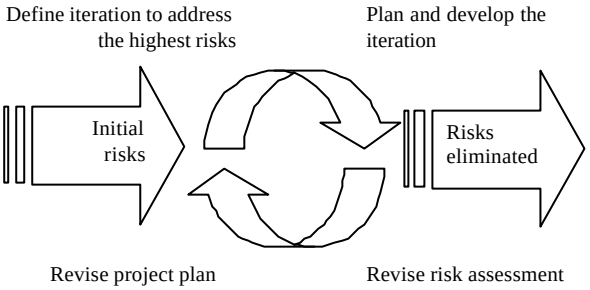

Figure 3 Iterative and incremental development

The design and development of safety critical systems requires extensive efforts and resources and is thus very expensive. Dependable computers have therefore preferably been used in military, space and civil aircraft applications. To be useful within other areas such as automotive or train control applications we must look for other solutions where we strive towards conceptual designs where we take advantage of application inherent redundancy (application specific fault tolerance) and where we can implement our designs with mostly standard, non-expensive electronic devices.

3. TRAIN APPLICATION CHARACTERISTICS

In a train, we might classify the fundamental objects according to their respective level of control as follows (figure 4):

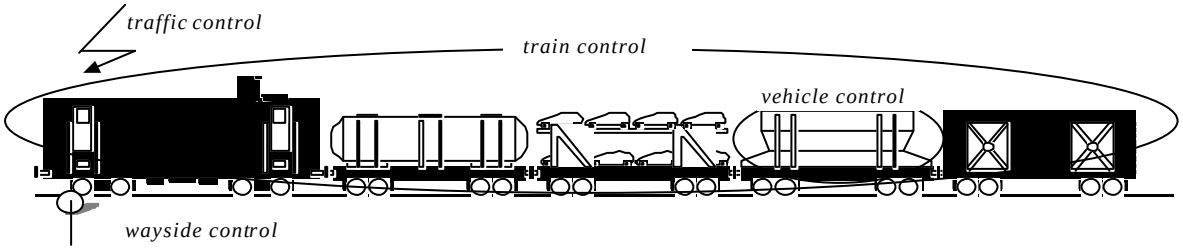

Figure 4 Levels of control

Train, which usually consists of a number of tractions and wagons and is the autonomous and controllable entity that moves along a track.

Traction, one or several traction units pull the train and today house the command and control system.

Wagon, also referred to as "car" or "carriage", is the fundamental, payload-carrying entity in the railway system. There must be a communication system in a wagon, which has the responsibility to communicate with other objects (wagons and tractions) as well as with subsystems in the wagon.

Traffic control system, also known as ATC, which encompasses all the immobile systems. It has the responsibility for route planning, navigation, signalling, surveillance and safety. It keeps track of all trains that might interact, their current position and destination.

Wayside control, a set of systems primarily intended for surveillance and control, communicates wirelessly with the traction.

Train command-and-control system (or simply "train control"). It has the responsibility for navigation, safety and surveillance. It controls the speed and has to keep track of actual values of allowed distance to go, braking distance and other dynamic properties of the train.

Vehicle control system, located in a car, encompasses overall responsibility for communication with traction and other cars. Controls and synchronizes all computer based systems in the car.

Vehicle control subsystem, several types of computer-based subsystems can exist in a railway vehicle and in a not too distant future, there will be a number of them. Examples for engines are drive control and braking. Examples for wagons are braking, lights, comfort systems and surveillance.

The use of computers in control systems brings, through their flexibility, a huge potential for new, innovative and cost efficient technical solutions. A future train car may be designed as a 'smart car' (see figure 5) with capacity of autonomous behaviour and its own integrity. The 'smart-car' abstract concept is introduced to denote an entity, carrying a load and at the same time keeping track of information regarding the load. This information, available at any time, may be used for example by logistic systems to track individual cargos. As another example, with the same information it should be possible to rearrange a train set practically without human intervention on a rail-shunting yard.

The 'smart-car' application itself has both interesting and challenging features. It shows a broad range of functionality, a wide range of requirements with different criticality that has to be integrated into a single unit working under varying conditions.

In the remainder of this chapter, we will discuss the 'smart car' in terms of suitability for the implementation of a vehicle control in general terms of dependability.

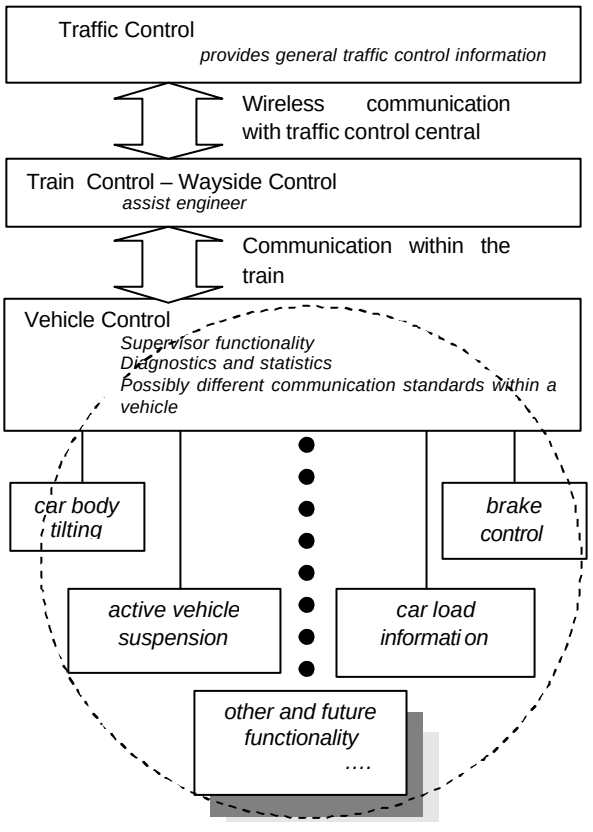

Figure 5 'Smart car' system structure

3.1 Vehicle control system

In Figure 4 at least three levels of application control were implied. These levels may also be described in a layered system structure where each layer is depicted to a corresponding control level and is designed to handle a finite set of system activities (Figure 5). Neither *train*, *traffic* nor *wayside* control is within our primary objectives in this work. However, the reader should be aware that the implementation of a future `smart car´ relies heavily on a train control system adapted for these new cars.

A train car should be equipped with a vehicle control system. A vehicle control supervises and controls one or several vehicle control subsystems.

A modern train car houses several applications suitable for computer control such as car body tilting, active vehicle suspension and brake control. Such applications are generally implemented as separate *subsystems* in the car. Figure 6 below, outlines an organization that might be used to house present and future types of subsystems. The figure shows how a vehicle is connected via a double bus to its closest neighbors (previous vehicle and next vehicle). The vehicle itself is always equipped with a vehicle control computer and, for reasons that will be discussed below, a redundant unit. Various types of vehicle subsystems are connected to and mastered by the vehicle control. Subsystems are assumed to be

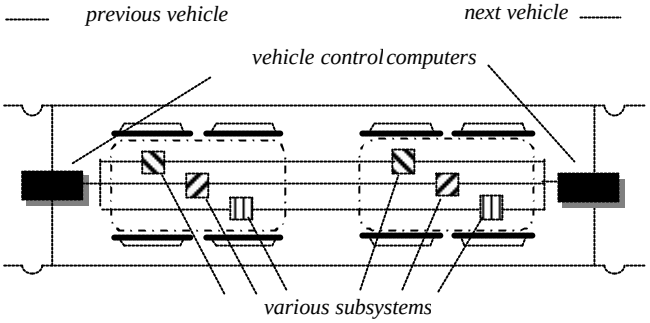

Figure 6 Vehicle control system

vendor supplied in general and there has to be a set of standards, describing the communication interface between the vehicle computer and its subsystems at different layers (in fact, such standards already exist in the automotive industry [5]. Subsystems should be interconnected as well as connected to vehicle control using at least two standard bus interfaces, for example UART and CAN. Depending on safety criticality, the subsystem might then be implemented by a manufacturer according to safety requirements and vehicle manufacturer requirements.

The vehicle control gets parameters from the train control system. The semantics of these parameters are sufficient to determine each car's behavior in different situations, for example:

- Each brake unit could include individual intelligence according to its load, position in the train etc. Such a system, presuming it is sufficiently dependable, will remove the needs for several of today's pneumatic and hydraulic systems, leading to a cheaper solution by reducing the number of system components.
- A vehicle control could provide for autonomous behavior in the event of extreme situations. For example, by monitoring the push and pull forces applied to the wagon from the neighboring wagons in the train, the local brake system could be made adaptive even in lack of train control.
- The vehicle control should also monitor each subsystem and provide diagnoses that can lead to rationalized maintenance-efforts, and decreased costs.

Both train and vehicle controls are subject to different types of requirements, such as *functional, timing* and *dependability* requirements. Although dependability requirements may be similar to both, we might expect large differences in functional as well as timing requirements. To further elaborate both train control and vehicle control functions we might anticipate a basic set of operational modes.

Modes of operation are recognized mainly based on which *normal state* the system currently is in. Additional modes of operation should be defined in order to handle *exceptional system states*. Exceptional system states are introduced to handle error situations where a system failure should be prevented, thus keeping the system in a safe state. Clearly different types of subsystems would react differently on a particular exceptional state and the following characterizations are thus far from complete.

The system might be in one of the following normal states depending on functional activities:

- *Idle, parked* - vehicle control is powered off

- *Idle, configuration* - vehicle control is powered on, train control system might inspect and configure all vehicle control systems.
- *Idle, ready* - train is inspected and configured, all systems are operational, all subsystems have potential of local control.
- *Idle, ready, degenerated* - train is inspected and configured, some systems may be non-operational or lack potential of vehicle control.
- *Moving* - all systems fully operational
- *Moving, degenerated* - performance is sub-optimal since all vehicles cannot be fully manoeuvred by train control

Note that the 'degenerated' state is not exceptional in this context. It merely denotes a train set where some (not all) of the cars lack the facilities of vehicle control. In such situations, a train control cannot be expected to deliver *optimal* performance. The degenerated state then implies that train control should deliver services at a *least required performance* level.

3.2 Safety issues in the vehicle control

When the system is exposed to a hazard, an exceptional state may be entered. Safety hazards may be observed at different levels. For example, if the engineer endangers the train by taking unpredicted (abnormal) action or should fall asleep, there would be an obvious risk of an accident. Such hazards are generally within concerns of train control. At vehicle control level hazards should solely be associated with faults in the system hardware and/or software. This should hold regardless of their nature, transient or permanent. We cannot, however, suffice with this observation. For example, a faulty neighboring vehicle control could forward unreasonable parameters from train control, thus causing an otherwise functional vehicle control to fail.

In [6], Kennedy discusses risk management and assessment for rolling stock safety cases. He demonstrates the ALARP (As Low As Reasonably Practicable) principle. A risk has to be demonstrated to either lie in the negligible range or if in the tolerable range, be ALARP. If we apply the same principle at vehicle control then it is sufficient to demonstrate that the safety performance does not fall outside the tolerable region. Consequently, standard methods for dependable computing can be used to demonstrate compliance with safety requirements in the vehicle control and the fault tolerance requirements could generally be stated as *Fail Operational/Fail Safe* (FO/FS) meaning that a device should continue deliver services as required despite the presence of a single fault. When a second fault occur the device should act safe i.e. the device should enter a pre determined fail safe mode defined so as to insure that no severe consequences could arise as a result from the fault.

A vehicle control may fail in several different ways giving rise to different *failure modes*. Each of these failure modes may expose the entire system to a smaller or a larger risk. The general approach is that any single failure mode should never expose the entire system to a risk that endangers system safety. Thus, we must assure that any single vehicle control failure mode should never propagate, nor give rise to a faulty behavior from a neighboring control.

A vehicle control can be functional or faulty possibly as determined by a common agreement within the system. A wide range of faults may occur and a certain range of faults have the potential to cause *commitment errors* among its neighbors, for example, a faulty communication device may deliver different status information to its neighbors or the communication medium may be corrupt. In either case, the sending vehicle control will appear to deliver inconsistent information to the surrounding (see Figure 7 below).

The situation may arise as result of a *transient fault*. If so we might attempt to handle the situation with software in the neighboring nodes, for example by executing a communication protocol which exchanges the message several times, providing several replicas of the same message and take a majority decision to determine the correct message. This strategy uses time redundancy and may be unsuitable for applications with tight real-time requirements. Furthermore it would not work in the case of a *permanent fault* since the faulty communication device should either deliver the same erroneous message ("stuck at - error") or deliver garbage unsuitable for a majority decision in the neighbor. Some strategies involve

message signatures providing means for error checking as well as error correction in the receiver, however such methods introduce considerable overhead which generally make them unsuitable for real-time applications.

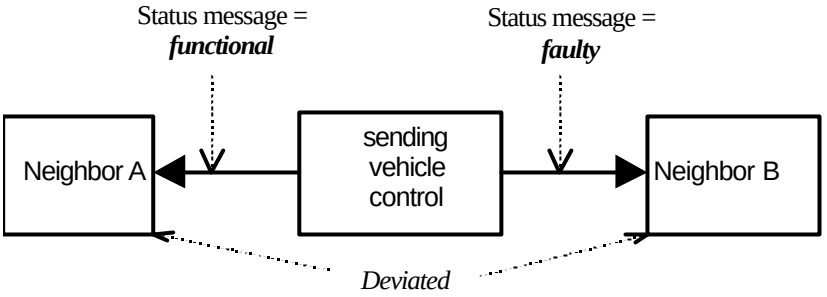

Figure 7 Commitment error

Commitment faults are serious in that they might potentially introduce a large number of *failure modes*, i.e. they may show up in numerous shapes. For these reasons we should not allow any form of commitment errors i.e. we require that a faulty vehicle control is always detected by its neighboring vehicle controls as well as the train control. We also require that such an agreement should be established within a fixed time interval after the fault has been observed for the first time. Consequently, a vehicle control must be at least outward fail silent. (Figure 8)

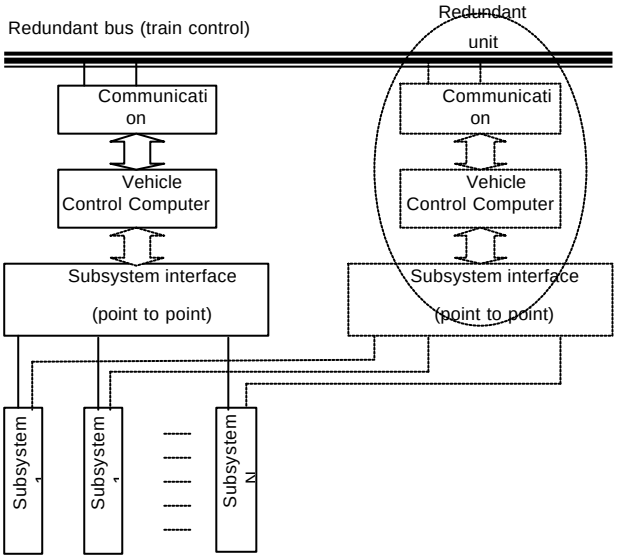

Figure 8 FO/FS vehicle control system

So far, we have focused on the vehicle control interface to train control and we have not really reached the interior functions. The essential question, from dependability aspects of view, is what fault tolerance requirements do we put on the vehicle control?

We should be aware that it is the overall safety requirement that leads to a FO/FS conclusion regarding the vehicle control. For example, a climate control computer for deep fried goods, may be designed fault-tolerant for a number of good reasons but probably not for the reason of satisfying any safety requirement. On the other hand, a brake system should be designed to provide at least fail-safe operation regardless of any failure mode in the vehicle control computer. In every design decision we must be aware of the fact that fault tolerant design is tedious, complex and expensive, so we do not wish to apply it unless it is strictly needed.

The final design of a vehicle control computer system may or may not arrive at a fault-tolerant solution due to the functionality that is actually allocated to it, i.e. safety requirements stated at "train control level" might be fulfilled by carefully designed, more or less autonomous subsystems.

Now, assume that the various subsystems implement fault tolerance according to their criticality and that all safety critical functions are maintained by each subsystem. The vehicle control computer functionality is now restricted to command and control autonomous subsystems.

3.3 General requirements applied to vehicle control subsystems

There is a broad range of requirements placed on a vehicle control as well as its subsystems. In this section, we will list the most important ones and then try to map them onto computer hardware and software requirements.

Functional, performance and dependability requirements (commonly referred to as RAMS, Reliability, Availability, Maintainability, and Safety) have traditionally evolved from national legislations and standards. More recently great efforts have been made towards a commitment to a common European standard. These efforts, performed by CEN (*European Committee for Standardization*) have this far resulted in a set of standards, (ENxxxxx) as well as draft standards (prEN), thus, this work is in progress. Document EN50126 [7] defines a process for the specification and demonstration of dependability requirements for the railway industry. It aims to "promote a common understanding and approach to the management of dependability". Other standards detail performance requirements as well as methods of tests. Such standards state safety requirements in terms of failure consequences. Precise measures and details concerning the formation of the train (the ordering of cars) are left for consideration by Transport Authorities, i.e. they are determined on national basis, see for example [8]. As a rule of thumb, however, for safety related functions the standard expresses that "...performance following any single failure is not less than required". The reader should observe that interpretation of this statement depends entirely on the observed system level. For example, a failing brake actuator may not cause any degraded performance in a large freight train consisting of several cars. However, the same failure might cause a disaster if it appeared in a tram.

A single car may be made out of parts from several different sub contractors. From the car manufacturer this, of course, is preferable, since the competition between several different sub contractors is assumed to produce parts of the highest quality at less expensive price. Moreover, a single sub contractor may not be able at all to provide different subsystems for all functions. At the same time, this might cause difficulties when integrating parts in a system. There is a prompt need for some sort of standardization of "in-vehicle" structures, organization, and interfaces. The vehicle control system has to be modular and flexible and adapt to safety critical as well as non safety critical subsystems.

Monitoring, *diagnosing* and *statistics* are examples of functionality that applies to the vehicle control itself as well as its subsystems. Perhaps the vehicle control functionality is restricted to forwarding messages between various subsystems and the train control but there might also be cases where these activities are initiated from the vehicle control computer. Here again, we recognize a need for standards or recommendations originating from vehicle manufacturers.

The implementation of hardware and software is always a trade-off between functionality, costs, time to market and RAM (Reliability, Availability, Maintainability). Furthermore, it is a common experience that newly designed software exhibits much more design faults than hardware does. In the following sections, we will briefly discuss hardware/software requirements emanating from safety-critical application requirements.

Hardware

Because it is used in safety critical applications, hardware should basically be designed fault-tolerant. Fault tolerance relies on redundancy, which tends to give complex and expensive hardware. However, for a range of vehicle subsystem control functions there is clearly a potential for application level fault tolerance that can be utilised to meet safety requirements. For example, in a brake system, we might exploit the fact that each wheel is equipped with a single brake actuator unit while the safety requirements apply to the entire brake system, which consists of several actuators. Besides safety

aspects, we must consider the economy in terms of development, production and maintenance costs. As an example, fault tolerant designs are often complex, application specific and dedicated and rely on the use of high quality and sometimes custom designed components. To find a cost efficient solution we would prefer to use standard components of commercial quality. At this point we meet the challenge where we have to define a hardware architecture that provide basic dependable services with suitable well defined fail-safe states and above this, a potential for the implementation of strategic intelligence by means of software. The architecture must be realistic to implement mainly by use of standard components.

Software

Use of software is a cost efficient means to fulfil requirements on flexibility, modularity, monitoring functions and diagnosis. Carefully constructed software also has the potential of increasing system safety by taking appropriate actions as response to exceptional events (failures). First of all we may consider some general requirements:

- *Efficiency*: Computer applications in general are cost sensitive. Hardware resources have to be utilised efficiently.
- *Real-time support*: The software has to support real-time requirements. There is a broad spectrum of timing requirements ranging from milliseconds to a few minutes. Time-triggered events as well as asynchronous events should be supported not only by blocking primitives such as semaphores but also expressed as real-time requirements.
- *Modularity and Maintainability*: Software should be structured and partitioned in a way that supports modular development and test. Modifications and extensions to existing software should be possible for example by allowing for replacement of a software module with a later revision. Such changes must not affect other functions if this can be avoided.
- *Reusability*: Software is commonly reused in different control systems and different projects. Thus the functional implementation should be independent of timing aspects and properties such as scheduling strategy.
- *Diagnoseability*: It should be possible to record and extract information about exceptional events (errors and failures) during normal operation.

When we develop software for distributed applications we must also consider the distributed software structure and allocation. The following normally calls for support from the implementation language:

- *Transparent/Forced Distribution*: In many cases the distribution of the software (between the computer nodes) may be hidden for the programmer. In other cases, however, there is an obvious allocation of tasks to specific nodes. Thus, an implementation language that provides semantics for a distributed architecture would be preferable.

We might want to consider issues related to software robustness:

- *General/Application defined exception handling*: The implementation language should also include semantics for differentiated exception handling. Clearly one can identify kinds of errors that should be handled at system level but there are other kinds of errors that can be efficiently handled by the application itself.
- *Predictability*: It should be possible to analyse an application with respect to functional and temporal behavior at compile time. Static analysis may require the omission of common programming constructs such as recursion and undetermined loops. However we must also consider more subtle programming constructions (hidden recursions such as A calls B calls C .....calls A). Temporal behavior is derived from (among other things) hardware characteristics and we may presume that analysis tools as well as simulators will become vital components in the program development environment.
- *Fault Tolerant Software*: Despite all efforts made during the program development process, unfortunately there is still a reasonable risk that faults exist in the software. We might require that redundant (diversified) software is executed, at least for our safety critical applications.

Choosing a good software implementation language is an essential step towards development of software for safety critical applications. The Ada programming language was designed for such purposes. However current Ada implementations often require powerful microprocessors and huge amounts of primary memory. This is not always the case for embedded control systems and it seems, current practices prefer to use the 'C' programming language or perhaps its object oriented ancestor 'C++'. Unfortunately, 'C++' suffers from similar restrictions as Ada as it requires large run-time libraries as well as huge memories.

The 'C' programming language has been selected for a wide range of real-time embedded applications within the automotive industry although the language itself allows program constructions that invites the programmer to ambiguous and error prone programs. The MISRA (*Motor Industry Software Reliability Association*) has tackled the problem by defining a restricted subset of 'C' called MISRA-C. Although this is not an attempt to promote the use of 'C' for automotive applications, it seeks to promote the safest possible use of the language.

4. CASE STUDY: THE BRAKE SUBSYSTEM

As an example of a vehicle subsystem with safety requirements, we now turn to a discussion about the interior of a brake control.

Before entering the discussion about electronic train control, we would like to recapture some essentials from historical and contemporary techniques and technology.

Today's train control, when discussing freight trains, is essentially a matter of braking the train. Current techniques often rely on air pressure as the information carrier as well as the switch functions that activate the physical brakes that applies physical pressure to the brake discs. This is an old technique that has proven to be very reliable but with increasing demands on train sizes and carrier loads this technique falls short depending on its physical constraints.

In order to release a car brake the pressure is pumped up to five Bars in the reservoir. The pressure is however decreased throughout the train due to leaking valves and may fall below the level required by the car brake to release. Thus, the length of a train set is physically constrained by this technique (Figure 9).

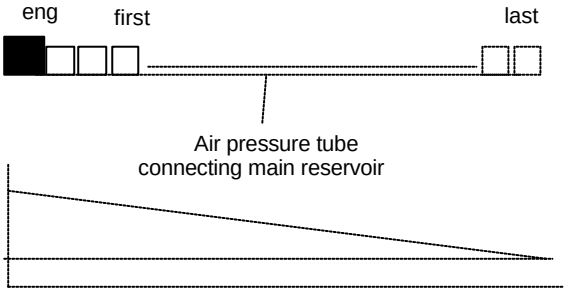

Figure 9 Air pressure drops with distance to car

In order to brake the train set, the engineer opens a valve causing air pressure to drop. When reaching a level of approx 3 Bars in each individual car, the car's brake is applied to the wheels. Thus, from simple physics, the first car will come first and the last car will apply brakes last.

From this very brief description of today's brake systems, we identify at least two serious drawbacks:

- Train set length, today limited to about 750 meters and a maximum of 40 cars. This might be compared to new limitations stated in INTELFRET[1] where a maximum train set length of 2250 metres, a capacity of 128 cars and nominal freight train speed 160 km/h, is proposed.

[1] INTELFRET - EU-project, intelligent freight trains

- Today's method of braking a train set is clearly far from optimal. Substantial economic gains and far less environmental impacts might be achieved with a brake system that takes the train set dynamics into account when braking the train.

The FEBIS- (*Freight Electronic Brake and Information System*)[2] [9] is currently developing ideas and concepts for a new generation of electronic systems located in engines and cars. These systems communicate digitally via electrical buses that connect local systems.

This new electronic platform provides entirely new capabilities and a range of new possibilities. In this next generation system, train control is initiated from the command and control systems which are, first of all, designed to assist the engineer. Global control is able to gather information such as train car characteristics and current payload about each car in the train from the local control systems. It is capable of calculating parameters used to describe the car's dynamics from a global point of view. The parameters are calculated depending on the current train set e.g. number of wagons, payloads, types of wagons etc. Characteristics may vary depending on the wagon itself, on the wagon's position in the train but also on other wagons in the train. Global control is able to supply each wagon with such parameters thus providing means for an individual adaptive local control.

Brake system functions

The brake system provides several functions. Basic requirements depend on the vehicle type, i.e. demands on freight car brake systems differ significantly from demands on a Light Rail Vehicle (LRV) or tram. Although some parts of the following discussion do not really apply to freight cars, we do treat them here in order to keep our discussion as general as possible.

Service braking is frequently used by the driver in controlling the train. Service braking shall achieve specified levels of performance at any time.

Emergency braking may be initiated by the driver or even by a passenger in the case of extreme hazards. Emergency braking shall achieve a specified level of performance and a high level of integrity.

Security braking is a particular form of braking. Security braking is activated in the case of system failure within the ordinary brake system, thus it is a redundant back-up system. The security brake system is designed to apply maximum forces, so as to stop the car within shortest possible distance. Security brake systems are often implemented using mechanical or electro-mechanical devices. A major advantage with such solutions is the high reliability accomplished with these devices. The serious drawbacks are the large size and high costs associated with this back up. From the contents, it is obvious that security braking shall achieve a high level of integrity.

Holding brake is a short duration brake used for ensuring against moving a vehicle once stationary, e.g. for un-load and load of passengers.

Parking brake should be able to hold a defined load on a defined gradient for an infinite time. It is intended for use while the train is stabled. The parking brake should be designed to ensure that it will automatically secure the train in the event of loss of emergency or service brake. On newly designed trains, the parking brake should apply automatically to ensure the security against movement of the train.

Wheel slide protection is fitted to optimize braking performance and to provide protection against wheel set damage e.g. during braking in poor adhesion conditions. Such systems are furthermore designed to minimize the braking force so as to achieve minimal practical stopping distance.

5. DISCUSSION AND FUTURE WORK

Railway transportation is an international matter. Trains and cars cross borders daily and a single car may be shifted among different trains during its journey to the destination. This is certainly an area where international agreements, standards and certifications are needed.

---

[2] FEBIS, a joint project between French and German National Railroads (SNCF and DB AG), for the design of a train control communication system that may supply all vital information needed for a computer based brake system situated in each vehicle.

A system such as a train carriage computer might soon become rather complex. It should provide for integration of subsystems from different vendors still providing dependable services. Each subsystem shall be maintainable in terms of various plug-in solutions. For example, a basic brake system adapts to extreme requirements through special software. A programmable device provides attractive flexibility, adaptive brake control is capable of compensate for worn-out mechanical parts as well as optimize wear which will lead to less environmental pollution and lower maintenance costs.

Another interesting possibility that emerges from the computer-based control is diagnosability, i.e. test programs that monitor different functions and establish status of mechanical and electrical parts in the brake system. Such facilities can be used to improve and optimize maintenance, which will lead to reduced costs and improved reliability.

The general train control application is naturally distributed, i.e. a set of carriages, possibly several tractions each comprising several different functions. A centralised system would soon become rather complex, probably be more vulnerable to disturbances, and require a tremendous amount of cabling. A distributed computer control system further provides a higher degree of hardware redundancy as well as means for a higher degree of software redundancy from the extra processing capacity added by the redundant microprocessors.

Functions controlling the train carriage dynamics are generally considered safety critical, in that failure to comply with the specified behavior might expose the entire system to great danger such as derailment. System safety standards are also subject to national legislation, for example, in Germany a non-mechanical-backup brake system in a car would currently be impossible since it is against the law. However there are several implications towards a change and this area is subject to large research efforts by several actors in the car supplier industry for the moment. In general, legislations concerning transportation are getting increasingly international in conformance with common standards and practices. A similar evolution is likely to be expected within the railway transportation area.

Today, computers are used for safety-critical control in space (i.e. space shuttle), in the air (civil and military aircraft) and on the ground (transport vehicles and automotives). To some extent, computers are also used in train carriages for passengers. Such systems are always combined with different types of mechanical backups to maintain safety in hazardous situations. Often these strategies result in electromechanical systems with massive and expensive redundancy.

We believe that during the years to come, technological innovations and scientific progress from the field of dependable computing will bring tremendous new opportunities for the railway industries.

We are currently working on a conceptual architecture for use in safety critical train applications in general. As a special case, we are developing these ideas for an all-electronic brake application, as we believe that these functions exhibit the most extreme dependability requirements. The architecture is suited for control applications where functional redundancy may be achieved at system level. The architecture will be further developed and demonstrated in a distributed control of braking systems in a train carriage. As a prerequisite, the architecture should allow for implementation using mostly standard components of commercial quality. The major advantages of this new architecture is:

- it is scalable in that it might be implemented using two or more nodes depending on the application.
- it provides basic fault tolerance of all transient and intermittent faults as well as at least one permanent fault
- the implementation relies on contemporary technology and might thus rapidly gain from new innovations in a revised implementation.

Our planned work for the near future aims at a full-scale implementation of a complete bogie brake system. We will furthermore elaborate on reliability calculations for all vital components as well as safety analysis of different configurations of the conceptual architecture.

Our goal with this work is to provide an open dependable distributed architecture for control applications in trains as well as methods for different kinds of evaluations of the architecture and its implementation.

Acknowledgments

Special thanks to Mr Jörgen Andersson of SAB WABCO AB for providing source material for this research as well as comments and suggestions regarding this paper. I am also grateful to Prof. Jan Torin and Mr. Håkan Edler for several valuable comments on earlier versions of this report.

References

[1] Coll, D.C.,Sheikh, A.U.,Ayers, R.G. and Baily, J.H. The communications system architecture of the North American advanced train control system. IEEE Trans. Veh. Technol.,August 1990, 39(3), 244-255.

[2] Ghosh. S, Fundamental issues in intelligent transportation systems, Proc Instn Mech Engnrs, Vol 213 part F,125-131, 1999.

[3] Haspel, U. Wigger, P. Safety aspects of driverless metros - assessment of the Copenhagen Metro, World Railway Management,1999

[4] Laprie, J,C. (ed.): Dependability: basic concepts and terminology (Springer Verlag 1991)

[5] Society of Automotive Engineers, SAE, Surface Vehicle Recommended Practice J1939-73.

[6] Kennedy, A, Risk management and assessment for rolling stock safety cases, Proc Instn Mech Engnrs, Vol 213 part F,67-72, 1997.

[7] EN50126,1999, Railway Specifications - The specification and demonstration of dependability, reliability, availability and safety (RAMS), CEN.

[8] B.O. Strab. ,Federal German LRT Construction and Use Regulations,1995, English translation

[9] Witke, Minde, Engelmann, Zentrale Komponenten eines Intelligenten Güterzuges, November 2000. ETR, Eisenbahn technische Rundshau.