A Maintenance-Oriented Fault Model for the DECOS Integrated Diagnostic Architecture

P. Peti, R. Obermaisser, A. Ademaj, H. Kopetz
Vienna University of Technology
Austria
email: {php,ro,ademaj,hk}@vmars.tuwien.ac.at

***Abstract*— The increasing use of electronics in the automotive and avionic domain has lead to dramatic improvements with respect to functionality, safety, and cost. However, with this growth of electronics the likelihood of failures due to faults originating from electronic equipment also increases. In order to tackle prevalent diagnostic problems such as the reduction of the fault-not-found ratio, a maintenance-oriented fault model is needed that serves as the basis for the classification of experienced failures.**

In this paper we introduce such a maintenance-oriented fault model that establishes the conceptual foundation of the diagnostic services of the DECOS integrated architecture. The fault model takes the component-based nature of today's distributed embedded systems into account. According to this model each experienced failure is classified according to the field replaceable units of the system.

***Index Terms*— Maintenance, Diagnosis, Fault Model, Integrated Architecture, No-Fault-Found Problem**

I. Introduction

There is a significant trend in the automotive and avionic industry to increase the number of electronic devices to provide functionality that goes beyond common mechanic/hydraulic subsystems. However, despite all the benefits it is important to state that with the increasing use of electronic devices in transportation systems the likelihood of malfunctions of electronics and thus the numbers of defective electronic components will also increase.

From a maintenance point of view the most important question is whether a replacement of a particular component will put an end to spurious system malfunctions. Today's onboard diagnostic systems typically do not adequately support the service technician in the fault isolation process. Thus, maintenance engineers have to rely upon incomplete and imprecise information. This lack of information often results in replacements of working components [1], [2]. Even worse, in many cases the faulty component remains unchanged. The resulting negative publicity and increased warranty costs are serious factors that influence not only the success of a product but also delay the introduction of emerging technologies. For instance, in the avionic domain the No Fault Found (NFF) problem is estimated to account for approximately 300 million dollars per year [3]. With an average cost of 800$ per removal of a single Line Replaceable Unit (LRU), there is a huge potential of cost reduction.

This paper devises a maintenance-oriented fault model in the context of integrated architectures in in order to reduce the maintenance associated costs. Integrated architectures promise massive cost savings by providing the possibility to share components among multiple applications in order to stop the increase of the number of deployed components. The integrated DECOS architecture [4] for dependable distributed embedded real-time systems is designed to overcome the "one function – one control unit" philosophy.

In order to tackle existing maintenance problems such as the NFF phenomenon, we propose a maintenance-oriented fault model that takes the component-based nature of distributed systems into account. Based on existing classification schemes [5], [6], [7], [8], [9] a refined Field Replaceable Unit (FRU) centered model is derived. This model establishes a basis for a better understanding of the diagnostic problems of modern distributed systems and introduces a maintenance specific fault classification. According to this model, the diagnostic analysis algorithms of the DECOS integrated architecture perform a classification of the experienced failures and anomalies in order to determine whether a change of a particular FRU can eliminate the experienced problem, or if a replacement (i.e. change of hardware or update of software) will prove to be ineffective. In case of distributed embedded real-time systems deployed in the automotive or avionics domain an FRU is defined as a component which is designated to be removed and replaced by line maintenance personnel, i.e. the service technician (see for instance [10]).

The paper is structured as follows. Section II describes the integrated architecture that is developed in

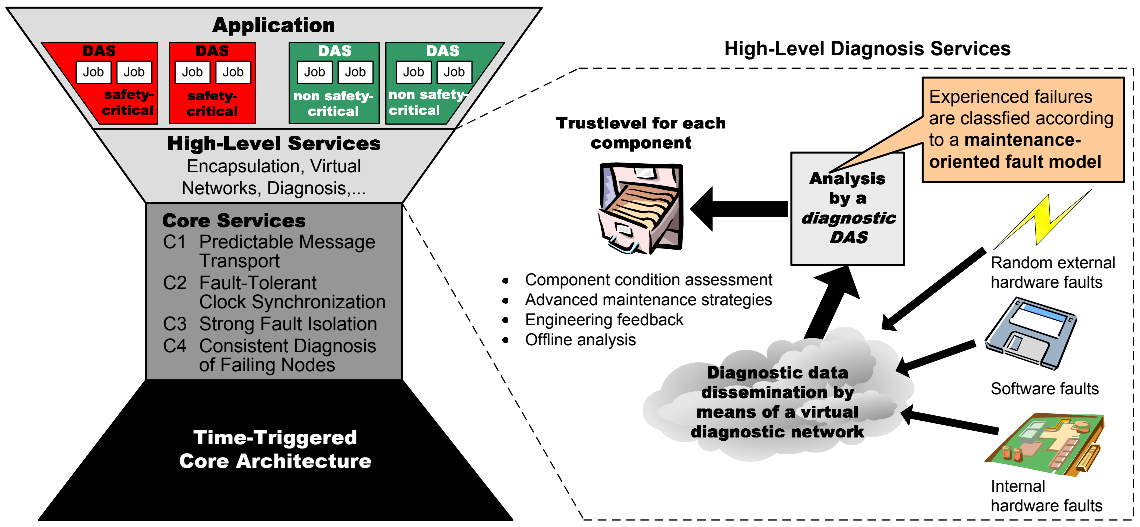

Application
DAS
Job
Job
safety-critical
DAS
Job
Job
safety-critical
DAS
Job
Job
non safety-critical
DAS
Job
Job
non safety-critical
High-Level Services
Encapsulation, Virtual Networks, Diagnosis,...
Core Services
C1 Predictable Message Transport
C2 Fault-Tolerant Clock Synchronization
C3 Strong Fault Isolation
C4 Consistent Diagnosis of Failing Nodes
Time-Triggered Core Architecture
High-Level Diagnosis Services
Experienced failures are classfied according to a maintenance-oriented fault model
Trustlevel for each component
Analysis by a diagnostic DAS
Component condition assessment
Advanced maintenance strategies
Engineering feedback
Offline analysis
Random external hardware faults
Diagnostic data dissemination by means of a virtual diagnostic network
Software faults
Internal hardware faults

Fig. 1. Integrated System Architecture

the Dependable Embedded Components and Systems (DECOS) EU Framework Programme 6. In particular, the integrated diagnostic architecture of DECOS uses the proposed model as the basis for the diagnostic assessment process to determine the health status of the FRUs of the integrated system. In Section III the maintenance-oriented fault model is introduced and mapped onto the DECOS field replaceable units. Section IV presents substantial evidence why the chosen fault model is suitable for classifying the prevalent fault types experienced in today's electronic systems. In Section V the determination of a maintenance action based on the fault model will be discussed. The paper is concluded in Section VI.

II. The DECOS Integrated Diagnostic Architecture

This section describes the DECOS integrated architecture for dependable distributed embedded real-time systems [4] and focuses on the integrated diagnostic services. The DECOS integrated architecture provides a framework with generic architectural services for integrating multiple application subsystems within a single, distributed computer system, while retaining the error containment and complexity management benefits of federated systems.

A. Functional System Structure

For the provision of application services at the controlled object interface, the services of a real-time computer system are divided into a set of nearly-independent subsystems, each providing a part of the computer system's overall functionality. We denote such a subsystem as a *Distributed Application Subsystem (DAS)*, since the implementation of the corresponding functionality will most likely involve multiple components that are interconnected by an underlying communication system. The implementation as a distributed system is a prerequisite for establishing fault-tolerance by redundantly performing computations at separate components that fail independently. In addition, the DECOS integrated architecture groups DAS with the same criticality into subsystems (e.g., safety-critical vs. non safety-critical as illustrated in Figure 1).

In analogy to the structuring of the overall system, we further decompose each DAS into smaller units called *jobs*. A *job* is the basic unit of work that employs a *virtual network* for exchanging information with other jobs, thus working towards a collective goal. The access point of a job to the virtual network is denoted as a *port*. Every job has access to its relevant transducers, either directly via the controlled object interface or via a virtual network of known temporal properties.

B. Waist-Line Architecture

As depicted in Figure 1, the integrated DECOS architecture is based on a time-triggered core architecture that meets the safety requirements of ultra-dependable applications. The core of such an integrated distributed architecture for time-critical systems must provide four *core services*: predictable transport of messages, fault-tolerant clock synchronization, strong fault isolation, and consistent diagnosis of failing nodes. Any architecture that provides these core services can be used as a base architecture for the DECOS integrated system architecture.

Based on the core architecture, *high-level services* such as a virtual network service as the communication

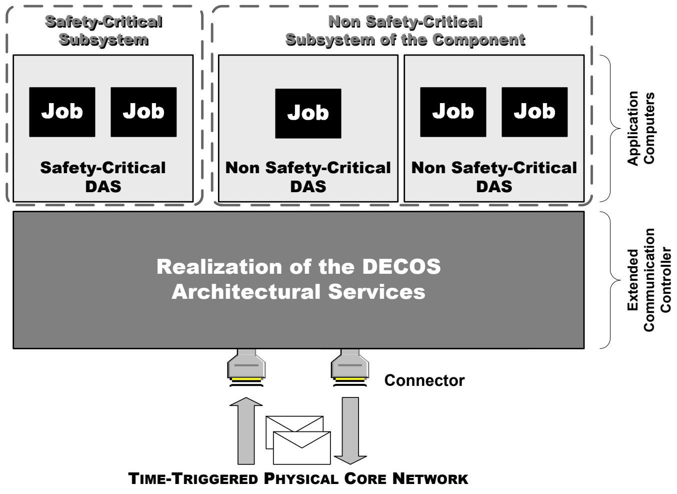

Safety-Critical Subsystem
Non Safety-Critical Subsystem of the Component
Job
Job
Job
Job
Job
Safety-Critical DAS
Non Safety-Critical DAS
Non Safety-Critical DAS
Application Computers
Realization of the DECOS Architectural Services
Extended Communication Controller
Connector
TIME-TRIGGERED PHYSICAL CORE NETWORK

Fig. 2. DECOS Component Structure

infrastructure tailored to the needs of each DAS, an encapsulation service for ensuring inner-component error containment, hidden gateways for the interconnection of DASs to improve quality of service and eliminate resource duplication, a redundancy management service (e.g., voting), and the diagnostic service as illustrated in Figure 1 are deployed.

C. The DECOS Component

In the DECOS component model we distinguish between two kinds of structuring, *horizontal* and *vertical* structuring as depicted in Figure 2. The vertical structuring of the component provides two subsystems within a component. The *safety-critical subsystem* is an encapsulated execution environment for ultra-dependable applications. The *non safety-critical subsystem* offers an environment for those applications having less stringent dependability requirements. For these applications, emphasis lies on low-cost, flexibility and resource efficiency. The safety-critical and non safety-critical subsystems are established by means of *spatial* and *temporal* inner-component partitioning [11]. In the DECOS component illustrated in Figure 2, two non safety-critical Distributed Application Subsystems (DASs) and one safety-critical DAS are shared among the DECOS component. Each of these DASs comprises one or more jobs.

A DECOS component can be horizontally structured into two layers: the realization of the core and high-level architectural services as described in Section II-B by means of an extended communication controller and the the application layer. The latter is comprised by application computers hosting one or more jobs. Each job is executed in a dedicated partition and communicates with other jobs of the same DAS by utilizing the virtual network services via the port interface.

For a more detailed description (e.g., constituting hardware elements and software modules) and an analysis of this model with respect to certifyability, encapsulation and independent development aspects refer to [4].

D. Integrated Support for Diagnosis

The DECOS integrated diagnostic architecture is designed to meet the requirements imposed by industry with respect to diagnosis such as support for advanced maintenance strategies, intellectual property protection, detection of correlated errors, service technician assistance and assessment of fault-tolerance mechanisms [12]. The model of the diagnostic architecture as illustrated in Figure 1 can be divided into three consecutive steps. Once a failure or anomaly is detected by the detection mechanisms of the diagnostic services, a corresponding message is disseminated via a dedicated *virtual diagnostic network*. A virtual network is an encapsulated overlay network on top the time-triggered core physical network [13]. The high-level virtual network service ensures that strong fault isolation between virtual networks of different DASs is guaranteed. This way no probe effect at network level can be introduced [14]. The subsequent analysis of this information is located in an encapsulated *diagnostic DAS* in order to determine the nature of an experienced fault with respect to a maintenance-oriented fault model. The diagnostic DAS outputs a trust level for each component, that acts as the basis for the decision of the maintenance engineer on the question whether a FRU should be replaced or remain in the system. The maintenance-oriented fault model introduced in this paper forms the basis for any health status assessment of the FRUs by the diagnostic DAS of the integrated system.

E. Fault Hypothesis

The fault hypothesis defines the fault model with respect to fault tolerance. In the integrated system architecture, we perform a differentiation of Fault Containment Regions (FCRs) for hardware and software faults [15]. For hardware faults, we regard a complete component as a FCR, because a component will be implemented as a System-On-a-Chip (SOC) and contains shared physical resources (e.g., processor, power supply). The failure mode of a hardware FCR is assumed to be arbitrary. The failure frequency in case of permanent hardware failures is in the order of 100 FIT [16]. In case of transient failures a significantly higher failure frequency in the order of 1000-10000 hours is assumed.

For software faults, we regard a job as a FCR. The failure mode of a job is a violation of the port specification in either the time or value domain. In case of a failure in the value domain, the content of a message does not conform to its specification, while in case of a timing failure, the send instant of the message is incorrect.

III. The Maintenance-oriented Fault Model of the DECOS architecture

The purpose of a model is to develop a reduced representation of the world, that helps in understanding the problem domain [17]. Related fault models presented in [5], [6], [7], [9] are developed for the purpose of fault tolerance. A fault model for maintenance, however, aims at allowing a determination whether a particular fault affecting the system will require a replacement of a FRU. Thus, a fault classification is necessary, that allows deducing the adequate maintenance strategy from tracing back the fault-error-failure chain. In case of integrated architectures such a fault classification needs to include both component hardware and software module faults, since in integrated architectures a component is shared among multiple software modules.

In the following we elaborate on the constituting elements of the maintenance-oriented fault model we consider important in the context of the DECOS architecture.

A. Unit of Replacement

There exists a strong relationship in the DECOS architecture between the fault hypothesis (i.e. the fault model for faul-tolerance aspects) and the fault model for maintenance. While in the fault hypothesis the FCRs are identified, i.e. the hardware units limiting the immediate impact of a fault to the system [18] of the system, in the maintenance-oriented fault model the FRUs with respect to hardware faults are stated. Typically, there will be a congruence, since maintenance of faulty components is also a key aspect in establishing the required level of dependability in a faul-tolerant system.

Hence, in the DECOS architecture we consider a component as the FCR/FRU for hardware faults and a job as the FCR/FRU for software design faults.

B. Fault Classification

In the fault hypothesis the statements about the faults at system level are defined that may occur if a FCR exhibits a failure (system level). In case of a maintenance-oriented fault model on the other hand, a classification of the faults affecting a FRU needs to be specified (FRU level). Thus, the two models classify faults at different levels in the fault-error-failure chain [6] as illustrated in 3.

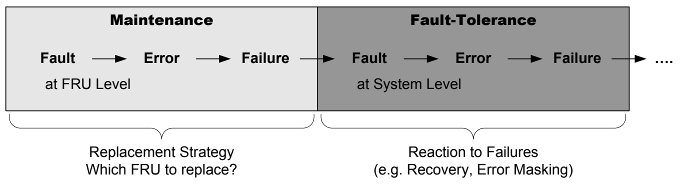


Fig. 3. The Fault-Error-Failure Chain

As stated in [15] the concept of fault is introduced to stop the recursion of the “fault-error-failure” chain. From a maintenance point of view, we are only interested in categorizing the type of fault of the experienced failure into classes that allow a determination whether a replacement is the correct maintenance strategy. Thus, by reversing the fault-error-failure chain [6], it must be possible for the diagnostic subsystem to determine whether a change of a FRU can eliminate the experienced problem, or if a replacement (i.e. change of hardware or update of software) will prove to be ineffective. On the basis of the maintenance-oriented fault model a corresponding maintenance action for each fault class needs to be stated.

Consequently, we stop the recursion at Field Replaceable Unit (FRU) level. In the context of the DECOS architecture in case of hardware faults the FRU is considered to be complete node computer, while for software faults the FRU is considered to be a job. The fault classification for each FRU needs to be derived by analyzing the prevalent types of faults affecting the given FRU.

Consider for instance a crack in a Printed Circuit Board (PCB). Such a crack may originate from wear-out of the material due to environmental (external) stress, such as vibration (e.g., rough roads), shock (e.g., chuckholes, hard landings) and changes in temperature (i.e. expansion and contraction). Depending on operational conditions this crack may cause the component to fail transiently. From a maintenance point of view (at the service station) the first level cause due to mechanical stress is not of interest. In analogy the exact element of the FRU that is subject to failure is of limited interest for a service technician. By taking a maintenance-oriented view the most important fact we are interested in is that the hardware fault can only be eliminated by replacement of the FRU. The analysis, which part of the FRU caused the malfunction is in the scope of the inspection of faulty nodes at the Original Equipment Manufacturer (OEM) (and not part of the maintenance action at the service station).

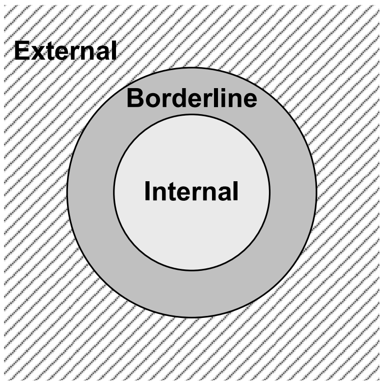


Fig. 4. Component Fault Model

C. The Component Fault Model

The model takes the component-based nature of today's distributed systems into account by considering a component as a FRU for hardware faults. Consequently, we devise the following fault classes as illustrated in Figure 4. Faults that originate outside the component boundaries are denoted as *external faults*. External faults are characterized by having no permanent effect on the functionality of the component. A restart of the component with subsequent state synchronization is a typical strategy to restore a correct state. An example for an external fault is Electromagnetic Interference (EMI) [19]. So-called *borderline faults* are the class of faults that cannot be judged to be external or internal with respect to the component boundary. An example for such a fault is a connector fault (a connector consist of two parts, one attached to the component, the other attached to the cable loom). Since this class is responsible for a significant number of system failures [20], we extend the boundary classification of faults as introduced by Laprie [15] by adding the class of borderline faults. Finally, *internal faults* cover those faults that originate from within the FRU boundary (e.g., crack in the PCB). In contrast to external faults, these faults can only be eliminated by a replacement of the component.

D. The Job Fault Model

In the context of integrated architectures, such as the DECOS architecture, a further differentiation of component internal faults is possible. While in architectures with the "1 Function - 1 Electronic Control Unit (ECU)" design philosophy such a differentiation is futile, in integrated architectures such a finer granularity is important for the discrimination between software faults and hardware faults.

The increasing complexity of software deployed in embedded systems requires the online diagnostic services of an architecture to provide means for identifying software design faults. An empirical study, although based on field data from the telecommunication industry, identified that only a small number of software modules is causing the majority of software related failures during operation [21]. If we act on the assumption, that a similar distribution of software faults is also feasible for the automotive/avionic domain, then a correlation of field data gathered by the online diagnostic services of a representative population provides a solid foundation for the identification of software design faults.

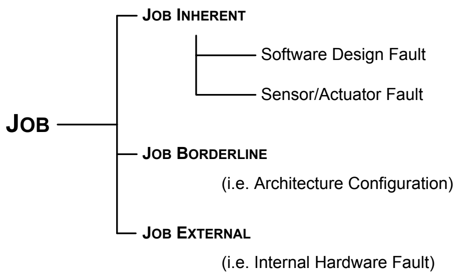


Fig. 5. Job Fault Model

As depicted in Figure 5 we discriminate for each job between job inherent, job borderline, and job external faults. *Job external* faults are faults affecting the internals of a component but do not origin within the boundaries of the job. In case multiple job external faults can be observed in one component, a component internal hardware fault can be assumed. Similar to the borderline faults at component level, *job borderline* faults are faults affecting the connectors, i.e. the ports of the jobs. Consider for example event-triggered jobs and ports accordingly. In case the jobs are operating as specified in term of sending messages according to an a priori defined probability distribution, there still might be the case where queue overflows occur (i.e. messages are lost). In this case a false configuration of the respective virtual network service is causing system malfunction. Job borderline faults are thus configuration faults. Finally, the class of *job inherent* faults are those faults that are originating from within the job. The class of job inherent faults can be further decomposed into *software design faults* and *sensor/actuator faults*. In the DECOS architecture each job is considered to have exclusive access to its sensors and actuators. Since, in general, one cannot differentiate by observing only the interface state whether a malfunction of the I/O hardware or a software design faults is causing unspecified job behavior, a differentiation of these two types is only possible by including job internal information into the assessment process.

Figure 6 gives an overview of the introduced maintenance-oriented fault model and relates the intro-

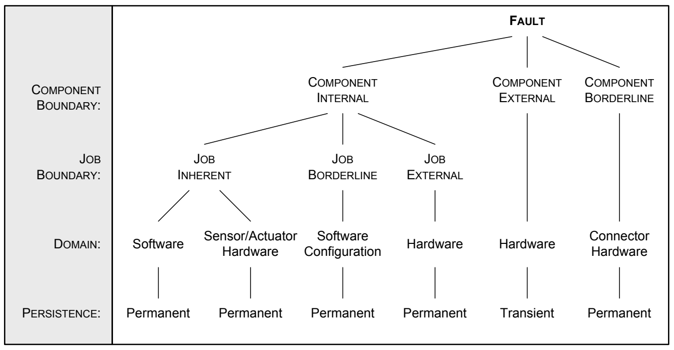


Fig. 6. Overview of the Maintenance-Oriented Fault Model

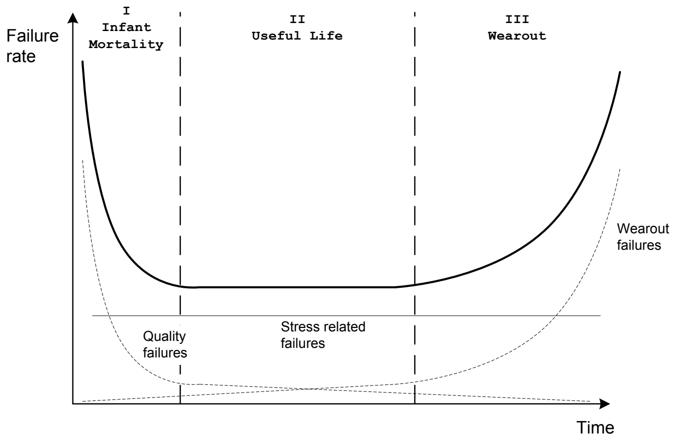


Fig. 7. Bathtub curve

duced fault model to the terms introduced in [6], [15]. The system boundaries are refined into component and job boundaries as the FRUs for hardware and software faults.

E. Assumptions behind the Fault Model

The reliability of electronic devices has been the subject of significant improvements in the manufacturing process by the IC industry. The permanent failure rate is very low compared to electronic components manufactured a decade ago [16]. The tremendous improvements in the reliability of semiconductor devices is reflected in *Pecht's Law*. It suggests that semiconductor device reliability in terms of time-to-failure is doubling every fourteen months based on activation energy trends of semiconductor devices [22].

The types and causes of failures for electronics have changed over the years. Failure analysis in recent years has revealed that some failure causes may have been reduced by improvements in technology but due to the higher level of complexity and downsizing other failure classes have emerged [23]. According to Constantinescu [24] the primary cause for the significant increase of soft error rates are shrinking geometries, lower power voltages and higher frequencies. These result in higher sensitivity to neutron and alpha particles, and consequently have an impact on dependability by increasing the transient failure rates. Furthermore, due to semiconductor process variations and manufacturing residuals the likelihood of component internal faults leading to transient failures is growing. The shrinking of geometries in semiconductor design has also significant impact on future design processes, such as nanometer design [25].

It can be summarized that the tremendous improvements made by the IC industry with respect to permanent failure are extenuated by increasing transient failure rates due to side effects of decreasing geometries of semiconductor technology.

As depicted in Figure 7 the reliability of electronic components can be illustrated by the bathtub curve [26, p. 5-28]. The bathtub curve is divided into three distinct phases, the infant mortality, the useful life, and the wearout phase. According to [27] infant mortality failures are typically due to mistakes made during the manufacturing process. Thus, improved manufacturing can significantly reduce the incidence of such failures (i.e. fault avoidance).

Based on field data from the automotive industry, Pauli and Meyna [16], [28] provide some very interesting facts on the failure rates during the period of infant mortality and useful life of the bathtub curve. In contrast to wearout failures that affect the entire population, infant mortality failures tend to affect only a subpopulation of the shipped product [27]. The reliability curve visualizes that the average failure rate of an ECU in the useful life period is very high. Reported failure frequencies are 50 out of 1 Million ECUs in 1 year. Harsh operating conditions with increased stress factors like temperature, shock and vibration, humidity, contaminants and radiation are affecting the reliability significantly [29], [30]. Such failure mechanisms due to accumulation of incremental damage beyond the endurance of the material are termed wearout mechanisms [31]. Unfortunately, the wearout period is typically not covered in statistics, since the manufacturers are only interested in field data during the warranty period of the product [16]. Wearout due to the continuous use and stress of components is a natural phenomenon. Consider for instance the break pads of a car. According to the time of operation and operating conditions the abrasion of the pads is more or less advanced. The same is true for the profile of a tire (e.g., on the landing gear of an airplane). For many non electronic devices there exists the possibility of visual assessment of the condition of the equipment.

Suitable indicators can be measured (e.g., depth of the remaining profile of a tire) and appropriate action can be taken by the service mechanics. The question raises, whether we can find suitable indicators in the domain of electronic devices that allow to effectively and undoubtedly assess the condition of electronic devices. If advanced maintenance techniques like Condition-Based Maintenance (CBM) are envisaged, then such indicators need to be identified [32].

A suitable indicator for wearout of electronic devices is the increase of transient failures in the system [24], [33]. In fact, these spurious failures need to be analyzed to distinguish between random external transient disturbances (e.g., due to EMI) originating from outside the LRU and transient failures caused by internal faults (e.g., solder joint cracks, loose contacts).

In order to allow the online diagnostic mechanisms to determine the type of fault that is affecting a particular FRU it is important to make quantitative statements about the underlying failure rates for both transient and permanent faults. In the following we present the assumptions behind the DECOS maintenance-oriented model:

- **Transient Hardware Failure Rate.** The transient failure rate of a FRU with respect to hardware faults is assumed to be in the order of 100.000 FIT, i.e. about 1 year. This failure rate is not well substantiated.
- **Duration of Transient Hardware Failures.** The duration of a transient hardware FRU failure can be assumed to be in the order of tens of milliseconds. For example in [34], the transient outage-time of an automotive steering system can be estimated as less than 50 ms.
  The time-triggered core physical architecture ensures that transient failures longer than the length of a slot of the Time Division Multiple Access (TDMA) round can be detected by other FRUs.
  In current automotive On-Board Diagnosis (OBD) systems, transient failures that are lasting for more than 500 ms are recorded. Failures with a significant shorter duration cannot be detected,
- **Duration of Correlated Transient Hardware Failures.** Correlated FRU failures, i.e. a fault affecting more than one FRU at the same time, are assumed to be experienced within a bounded interval of time. According to the ISO 7637 standard [35] the duration of an EMI burst is in the order of 10 ms.
- **Wearout Indicator.** In the DECOS model we regard increase of transient failures of a FRU as a suitable indicator for wearout of the electronic device [24].
- **Permanent Hardware Failure Rate.** The permanent failure rate of a FRU with respect to hardware faults is considered to be in the order of 100 FIT, i.e. about 1000 years [16].
- **Software Faults Distribution.** We assume that safety-critical jobs are certified to the necessary degree and thus free of software design faults. In case of non safety-critical jobs, we assume that a minority of the deployed software FRUs is causing the majority of software related failures during operation [21].

IV. Suitability Analysis

The following section is devoted to underpinning the suitability of the introduced maintenance-oriented fault model to differentiate faults experienced in real-world systems. We present an analysis that covers examples from literature for both the component and job classification.

A. Component Fault Model

1) Internal: In the following we will discuss component internal fault sources on the basis of representative examples found in literature.

a) Printed Circuit Board: The PCB interconnects the constituting hardware elements of a node.

- **Design.** A typical design fault with respect to PCB is erroneous layout. Consider for example faulty spacing between wires or incorrect placing of electrical elements on the PCB.
- **Manufacturing.** Manufacturing faults include all faults originating from the technical process of creating the PCB. Here, solder mask problems, defective vias or wrong assembly are among the typical faults. Also soldering related faults, such as defective solder connections, shorts or loose contacts fall into this category.
- **Operational.** Environmental stress factors (e.g., vibration, shock, humidity, chemicals) can lead to subsequent failure of electronic devices. Continuous exposure to these factors can result in cracks in the PCB due to wear-out of the board or over-stress resulting from thermal cycling or shock. The PCB is the primary source of component failure in case the ECU is exposed to stress conditions [29].

b) Discrete Elements: According to field studies [16], [29] resistors diodes and transistors have the lowest failure rates. Capacitors are an exception since these elements are more affected by aging processes and thus having a higher failure rate due to wearout that other discrete electronic elements.

c) Quartz: Since measurement of time in computer systems is based on frequency of oscillation of a quartz crystal, the correct functionality of these elements is of paramount importance. In case the system relies on precise timing information, environmental influences and wearout can have a significant impact on the drift of the clock. Consider for example systems where components are synchronized in order to achieve a consistent global timing information. Here, a defective quartz can cause a component failure due to loss of synchronization. As indicated in [29] the failure rate cannot be neglected. During operation the quartz can be influenced by many factors. Low power supply, thermal cycling and mechanical damage due to shock and vibration are probably the most common causes for permanent or transient faults [36].

d) Integrated Circuits: As indicated by the failure rates provided in [16] the higher the integration of the electronic elements, the higher the likelihood of failure. Therefore, integrated circuits are causing a significant number of component failures. In the following we will identify possible design, manufacturing and operational faults. An excellent overview on semiconductor faults can be found in [37].

- **Design.** The reliability of electronic devices was subject to significant improvements in the last decades. However, the downsizing of semiconductor features has lead to decrease in the gate oxide thicknesses and the distance between metallization, resulting in higher electric fields across the gate and possibility of failure, such as gate oxide breakdown and hot carrier damage [22].
  Due to the increasing level of complexity of modern integrated circuits the likelihood of design faults is non-negligible. For example Lev and Chao [25] state that, in nanometer design, wiring delay accounts for the vast majority of overall delay. Thus, the shrinking of geometries in semiconductor design has significant impact on future design processes. Besides wiring delay, cross coupling effects originating from incorrect design can result in transient failures during operation.
- **Manufacturing.** Due to semiconductor process variations such as intra-chip variances or mask alignment and manufacturing residuals the likelihood of reoccurring permanent faults leading to transient failures is growing [24]. For example consider a short of metal lines caused by an unexposed photo resist or a solid-state particle deposited on the metal layer before metal lithography.
- **Operational.** According to Constantinescu [24] the primary cause for the significant increase of soft error rates are shrinking geometries, lower power voltages and higher frequencies. These result in higher sensitivity to neutron and alpha particles, and consequently have an impact on dependability by increasing the transient failure rates [37].
  Another significant source of failure is variability in power supply and temperature effects. As stated in [38] temperature has strong influence on the properties of semiconductor materials. On the device level, mainly degradation and breakdown of oxides are the main cause of failure.

2) Borderline: Wiring and connector problems account for a significant proportion of electronic system failures. Several studies document the significant proportion of connector and wiring failures in distributed embedded systems. Field data cited by Swingler et al. [20] indicate that more than 30% of electrical failures are attributed to connection problems. Considering, that a luxury car can have up to 400 connectors this number underpins the potential for failures due to connectors in the automotive domain. In the avionic domain, Galler and Slenski identify interconnection problems as the major cause of aircraft electrical equipment failures [39] with a percentage of 36%. These numbers are underpinned by a study of the US Air Force reporting that 43% of mishaps related to electrical systems were due to connectors and wirings [40], [41].

The reliability of the interconnection system is of great importance for the correct operation of an electronic system. The physical interface between the electronic control units (i.e. the components) and the interconnection system remains one of the weakest links in terms of reliability, implying potentially catastrophic consequences [42]. For more information concerning the reliability of connectors the reader is referred to [43].

A significant problem regarding connector and wiring failures is the fact that the failure analysis or the testing procedure may itself be a corrective action for the source of the failure (e.g., such as when resetting a connector or when applying wipe to a connector electrical pad) [23]. The same is pointed out by [44], who states that analysis and repair is often difficult due to the possibility that any evidence of failure is inadvertently destroyed during extraction or inspection.

3) External: In the following we will discuss the most important external faults that have an impact on the functionality of the DECOS components. The class of external faults covers external influences such as cosmic radiation, temperature, EMI, shock, vibration, and humidity, only to name a few [30]. An extensive list of environmental factors having impact on the reliability of a component can be found in [26, p. 7-129].

a) Cosmic Radiation: In aerospace transient disturbances of components are frequently caused by Single Event Upsets (SEUs) originating from cosmic radiation. According to [31] such radiation induced failures in avionics are caused by uranium and thorium contaminants, and secondary cosmic rays. Among the consequences of radiation are aging effects (wearout), embrittlement of materials, and overstress soft errors in electronic hardware.

In [45] Single Event Upsets (SEUs) caused by cosmic rays in avionics are investigated. Based on the experimental evidence from measured in-flight occurrences of SEUs fault rates in dependence of the flight altitude are evaluated and the sources of such incidents identified. However, SEU are not restricted to higher altitudes as shown in [46].

b) Electromagnetic Interference: EMI imposes a serious threat to the intended function of electronic systems deployed in various application environments. In the following we will give a short overview of EMI related problems in the avionic and automotive domain.

One major source of EMI in the avionic domain is the effect of lightning on aircrafts. Besides severe effects on the aircraft skin (e.g., melting, deformation due to pressure waves), damage in externally mounted materials and vaporization of conductors, a serious consequence of lightning for the electronic equipment are the electric and magnetic fields. In [47] a 16.5% failure rate of electronic equipment of commercial airlines due to lightning strokes is reported. Since modern aircraft highly depend on the correct functionality of the electronic flight control system standards exist, to provide necessary aircraft protection [47].

Similarly, in the automotive domain the increasing use of electronics makes cars more susceptible to problems originating from EMI, and thus makes it a major consideration in vehicle electrical system design [48]. For example in [49] serious effects of EMI are mentioned, namely the unexpected shut off of car engines on highway overpasses. Another example for transient disturbances generated by EMI is noise of the ignition system of a car [50]. The UK based Motor Industry Software Reliability Association (MISRA) consortium has released guidelines for dealing with EMI in automotive environments [51]. The guidelines consider interference effects on various aspects of processing, for example communication lines, digital and analogue inputs, corruption of memory and loss of control of the processor. Similar impacts of EMI have been studied in [52].

A study on effects of electromagnetic interference on controller-computer upsets and system stability revealed that controllers are more susceptible to these unwanted noises due to shrinking device size, lower switching energy, and higher speed operations [19]. Typically, these external faults are likely to be transient and cause primary functional error modes in digital systems.

c) Environmental Stress Factors: Since transport vehicles, such as cars and airplanes, are usually exposed to harsh environmental conditions, the reliability of the deployed electronics is also exposed to this hard conditions [53]. Especially, climatic and mechanical stress decreases the lifetime of electronic equipment. For example, humidity, extreme temperature and moisture in combination with stress factors such as shock and vibration results in increasing wearout rates [54], [29].

Just to give an impressio, in the automotive industry temperatures can reach up to $200°$C on the engine or even $800°$C at the exhaustion system. Similar, the vibration and shock levels can reach up to 50g [38].

Thermal cycling, continuous vibration and shocks and environmental conditions like salt spray, dust or gravel that weaken the protection mechanisms (e.g., sealing, housing) are a serious threat to the reliability of of electronic devices deployed in electronic architectures.

Due to continuous exposure to environmental stress, external faults can be transformed into internal component hardware faults (e.g., continuous shock causes crack in PCB). Such an accumulation of incremental damage beyond the endurance of the material is termed *wear-out* fault [31].

d) Wiring: Wiring related problems are posing a serious threat to safety-critical systems. It is an acknowledged fact that every densely wired system is vulnerable to consequences of wiring problems. For instance Swissair 111 and TWA 800 have crashed because of faulty wiring [55].

According to [40] the aging process of wiring can be understood as degraded performance due to accumulated damage from long-term exposure. This includes damages resulting primarily from operational conditions, such as damages from chemical, thermal, electrical, and mechanical stress. Besides these stresses induced by the operational environment damages also originate from installation and maintenance practices. Such wiring failures frequently appear as broken conductors and damaged insulation which can be disrupt electrical signals and/or lead to arcing, that may have fatal consequences.

B. Job Fault Model

1) Inherent: The class of job inherent faults as introduced in III-D is divided into software faults and transducer (i.e. sensors and actuators) faults. In the following we will discuss why such a classification is feasible for today's distributed embedded real-time systems.

a) Software Faults: Gray [56] divided software faults into *Bohrbugs* and *Heisenbugs*. Bohrbugs are software design faults that deterministic in nature. In contrast to Bohrbugs, that can be identified during testing, Heisenbugs belong to the class of software design faults, that are very difficult to detect through testing procedures, since Heisenbugs are perceived as transient failures.

Although automatic code generation tools such as TargetLink from dSpace or the Real-Time Workshop for MATLAB/Simulink are increasingly becoming accepted in industry [57] in order to reduce software implementation faults [58] and speed up development, the increasing complexity of applications leads to an increased probability of software design faults. In particular, Heisenbugs remain frequently undetected and can only be identified by a fleet analysis during full operation of the product. For example, a software bug in an electronic management unit of the fuel pump caused some cars to stall if the fuel tank was below one third full. The resulting recalls not only impose a serious financial burden for the manufacturer but also have a significant impact on the reputation of the products.

In [59] the support of integrated diagnostics for software is underpinned by the provision of statistics indicating that 17% of the efforts associated with software maintenance are for correcting faults. Furthermore, 54% of the efforts associated with software support require an integrated set of diagnostic tools and techniques. The importance for the detection of software faults during system operation is stressed by stating the fact that despite all efforts to reduce software faults during development, there will still be latent software faults during testing and deployment (at least for non safety-critical systems). The issue of faults that can only be identified during operation is also raised in [60] in the context of the automotive domain. During product development and testing low quality issues are relatively easy to identify because they are uncovered with smaller sample size. The problem of current vehicle testing process is the identification of statistically very unlikely occurring incidents that become only identifiable after high volume production.

A recent study of software faults revealed that only a small number of software modules contain most of the faults discovered during pre-release testing [21] supporting the results of [61]. However, the discovery of these faults during pre-release testing is a very challenging task. In case of software faults detected during operation a distribution according to the *20-80 rule* has been identified, indicating that 20% of the software modules are causing 80% of the software related failures during operation.

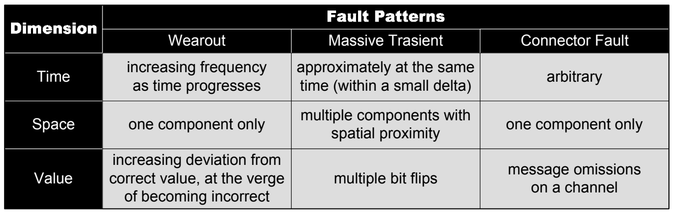
b) Transducer Faults: Sensors and actuators are the linkage between the controlling computer system and the controlled object. In the DECOS architecture each job is considered to have exclusive access to its sensors and actuators (e.g., electromechanical brake, window lifter, wheel speed sensor). An overview of sensors currently deployed in automotive industry can be found in [53], [62]. For the avionics domain the reader is referred to [63]. In the automotive domain the expected lifetime of sensors is assumed to be in the order of the lifetime of the car. For example in [64] the lifetime of automotive pressure sensors is specified between 10 and 15 years.

One approach to the highly application-specific diagnosis of job inherent faults is model-based diagnosis [65]. Based on a diagnostic model the application programmer transforms a model into application-specific assertions that are checked at run time. In [66] an example for model-based diagnosis in the automotive domain is presented. The author presents a diagnosis solution for the air-intake system of an automotive engine.

2) Borderline: The configuration of a distributed embedded real-time system is typically tool supported in order to minimize the possibility of faulty configurations (for instance see [67]). The class of borderline faults comprises those faults that emerge by deriving the configuration parameters on the basis of a communication model that is based on assumptions that do not hold in reality.

Consider for example an event-triggered legacy application. Temporal correctness of such an application can depend on temporal properties, such as bandwidth guarantees, bounds on communication latencies, and predefined message orderings. Furthermore, knowledge about the temporal behavior of communication activities is essential for the dimensioning of message buffers as required to tolerate temporary imbalances of message interarrival and service times [68]. If a subset of the assumptions of a legacy subsystem was made implicitly and not described in technical documentation, then determining a valid configuration is complicated. With incomplete knowledge about the assumptions that have been made by legacy applications concerning the underlying architecture, finding a consistent configuration becomes a non-trivial and error-prone activity. We denote any misconfiguration of the architectural services that results from incomplete knowledge about legacy applications as a borderline fault.

3) External: A job external fault can be mapped onto a component internal hardware fault. In case of



Dimension	Fault Patterns		
	Wearout	Massive Trasient	Connector Fault
Time	increasing frequency as time progresses	approximately at the same time (within a small delta)	arbitrary
Space	one component only	multiple components with spatial proximity	one component only
Value	increasing deviation from correct value, at the verge of becoming incorrect	multiple bit flips	message omissions on a channel

Fig. 8. Examples of Fault Patterns

a one-to-one mapping between jobs and components as in federated systems, this differentiation is obsolete. However, in the context of an integrated architecture such a differentiation is important to determine whether a component internal fault is a job inherent fault.

V. Determining the Maintenance Action

The introduced maintenance-oriented fault model serves as the basis for the assessment process as part of the DECOS integrated diagnostic architecture as indicated in Figure 1. Based on the classification of the experienced faults into the derived fault classes a particular maintenance action can be performed by the service technician.

A. Operation on the Distributed State

The pivotal strategy of the DECOS diagnostic architecture is the establishment of a holistic view on the system by operating on the *distributed state* established via the underlying core services. In combination with the strong fault isolation core service (C3 in Figure 1) and the encapsulation high-level service, this strategy allows to trace correlated component or job failures back to the FRU responsible for the experienced system behavior.

Whenever a fault affects one or more constituting parts of the distributed system, a change of state can occur that leads to an unintended state denoted as an error [6]. Depending on the type of fault (e.g., component external fault, job inherent fault), the unintended state will exhibit a characteristic manifestation in time, value and space. To capture the characteristics of the fault-induced distributed state changes, we introduce the concept of a *fault pattern*. A fault pattern is the set of state variables that has been identified as subject to fault-induced state changes along with corresponding properties in value, space, and time. Different types of faults show different fault patterns on the distributed state (see Figure 8 for examples).

In the diagnostic architecture so-called *Out-of-Norm Assertions (ONAs)* [69] are deployed that are checked against the distributed state established by the use of a sparse time base [70]. We define an Out-of-Norm Assertion (ONA) as a predicate on the distributed system state that encodes a fault pattern in the value, time and space domain. Out-of-Norm Assertions (ONAs) are deterministically triggered, whenever all symptoms of a particular fault pattern are detected on the distributed state. A *symptom* is a condition on a set of interface state variables of a particular component that is monitored to detect deviations from the Linking Interface (LIF) specification [71]. An ONA will likely be composed of more than one symptom, each operating on the interface state of different components.

B. Deriving Fault Patterns

Typically, defective control units are returned to the OEM for warranty analysis [72]. Although this off-line analysis is not in the scope of the DECOS integrated diagnostic architecture, the information gained through off-line analysis has a major impact on the design of the fault patterns. An optimization of the patterns in order to support the identification of those faults that have been identified to cause the majority of failures is of paramount importance to the effectiveness of the deployed diagnostic mechanisms. Studies of faults affecting ECUs used in automotive applications underpin the so-called Pareto-principle, i.e. a phenomenon that can have many theoretical causes has in reality only a few [16].

In order to derive the fault patterns for prevalent fault types causing the majority of system failures, a thorough analysis of field data (provided by industry) and fault injection techniques is necessary. Statistics on the types of faults affecting products in operation will help to derive those fault patterns that will help to identify the faults that will most likely affect the system (e.g., car, aircraft) during operation.

C. Determining the Replacement Strategy

The introduced fault model serves as the basis for the assessment process. The diagnostic subsystem executes algorithms on the gathered diagnostic information in order to assess the condition of each FRU. The evaluation process performed by the diagnostic DAS is illustrated in Figure 9. The evaluation process is based on a consistent notion of state, which is provided through the *action lattice* of the sparse time base established by the core services of the integrated architecture. The arrows in Figure 9 indicate the LRU assessment trajectories. At first both arrows show conformance with the LRU specification. As time progresses arrow A exhibits an increasing confidence for a violation of the specification, while arrow B indicates a LRU behavior in accordance with the specified service.

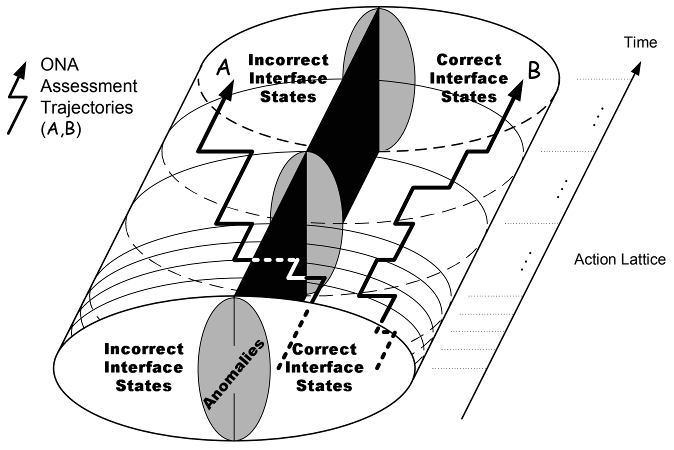


Fig. 9. LRU Assessment Process

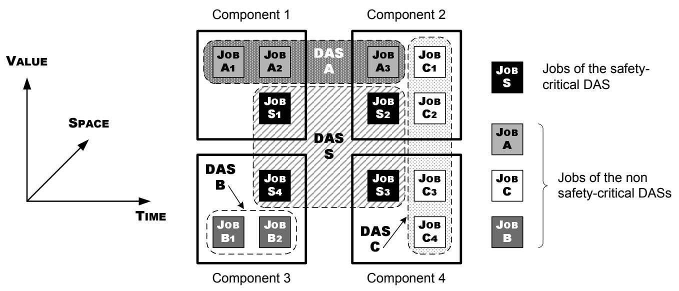


Fig. 10. Judgment According to the Three Dimensions: Time, Value and Space

The introduced integrated architecture provides a finer granularity of diagnostic information than federated systems. The assessment process exploits this knowledge about the functional and physical structure of the integrated architecture. The decomposition of the overall system into DASs with respective jobs is a key element for a more precise differentiation of experienced faults. By including the three dimensions of time, value, and space into the judgment process, a discrimination into the fault classes identified by the maintenance-oriented fault model is possible.

For instance, consider the system depicted in Figure 10. In case a job inherent fault hits the jobs A_1, A_2, and A_3 of the non safety-critical DAS A, the fault effects only the DAS A, since the error containment mechanisms of the architecture ensure that this fault cannot propagate to other DASs. In contrast, in case an internal component fault hits a component hosting multiple jobs of different DASs, it is very likely that the impact of this fault is not limited by DAS borders. An internal component hardware fault will cause multiple jobs hosted on one component to fail (e.g., the jobs A_3, C_1, C_2, and S_2 on component 2 in Figure 10).

The recognition of correlated job failures is also important in the detection of faults affecting architecture supported fault-tolerance mechanisms, such as Triple Modular Redundancy (TMR) mechanisms. This fault-tolerance mechanism is characterized by the replication of identical jobs on three different components in order to tolerate single hardware faults. In case the jobs S_1, S_2, and S_3 are forming a TMR system, the spatial dimension of an ONA covering deviations in the services of the three replicas spreads across components 1, 2, and 3 (since a component is the FCR with respect to hardware faults). In case one of the replicated safety-critical jobs fails, an analysis if correlated failures of jobs of other DAS executed at the same time on the same component exist will supply evidence whether a an internal hardware fault effects the component.

In addition, for the differentiation whether transient failures are caused by environmental influences or internal faults, techniques such as the α-count mechanisms can be utilized [33]. By interpreting the detected failures in the time and space domain, a determination between external and internal component faults is possible, since transient component internal faults tend to occur at a higher rate compared to transient component external faults and occur repeatedly at the same location [24]. This discrimination is of paramount importance since internal component faults can only be eliminated by repair, while a replacement of a component due to an external component fault will only increase the fault-not-found ratio (i.e. the component will be retested OK at bench tests).

As a result of the diagnostic judgement process, a *trust level* for each FRU of the system is determined that forms the basis for decision-making process of the maintenance engineer. Figure 11 summarizes the maintenance actions for each fault class:

- **Component External.** In the proposed model we consider the persistence of external faults as transient. Consequently, in case of component external faults no maintenance action has be to taken.
- **Component Borderline.** Borderline faults require a closer inspection by the service technicians. Connector problems, are difficult to trace, since the inspection itself can be the corrective action [44]. In case of connectors showing wearout phenomena such as fretting or corrosion, a replacement will be necessary.
- **Component Internal/Job External** Component internal faults such as a crack in the PCB or a defective processor require the replacement of the component (i.e. the ECU in the automotive domain or a Line Replaceable Module (LRM) in avionics).
- **Job Borderline.** Job borderline faults require the

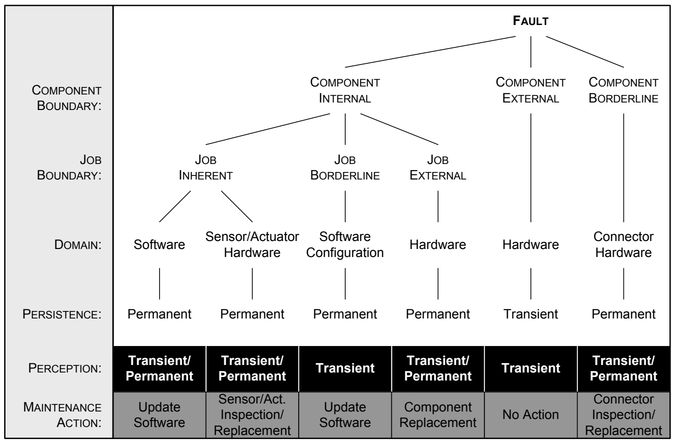


Fig. 11. Determining the Maintenance Action for each Fault Class

update of the configuration data of the virtual network service of the DAS.

- **Job Inherent.** Sensor/actuator faults require further inspection by the service technician in order to decide whether part replacement due to wear-out (e.g., change of brake pads) or a replacement of the transducer is necessary.

  Software faults identified by the diagnostic system requires an update of the job software in case this identification has also been acknowledged by the OEM and a corrected version of the job has been distributed to the service station. In case no update is available, this field data will be forwarded to the OEM in order to allow a correlation of the field data provided by a representative number of products to allow the identification of possible software design faults (i.e. fleet analysis as engineering feedback).

VI. Conclusion

Existing fault models developed for the purpose of fault-tolerant system design cannot be used unmodified for maintenance-oriented diagnosis, since maintenance requires a mapping of the experienced failures to field replaceable units, while fault-tolerance aims at keeping the system operational despite the occurrence of failures. In order to tackle today's prevalent diagnostic problems, we propose a maintenance-oriented fault model tailored to the emerging diagnostic needs of maintenance engineers and service technicians. The model synthesizes well-established concepts of distributed embedded real-time systems in order to form a conceptual basis and a foundation for solutions to prevalent diagnostic problems such as the fault-not-found phenomenon industry is currently facing. The classification can easily be mapped onto the field replaceable units of today's distributed embedded infrastructure as deployed in the automotive and avionic domain. The model distinguishes fault classes with respect to hardware and software faults and is thus in particular suitable for emerging integrated architectures.

Acknowledgments

This work has been supported in part by the European IST project ARTIST2 under project No. IST-004527 and the European IST project DECOS under project No. IST-511764.

References

[1] M. Mateos, P. Robin, S. Sauvage, V. Joloboff, G. Madhusudan, and Y. Bennani. Environment for evolutionary automotive diagnosis. In *Convergence International Congress & Exposition On Transportation Electronics*, Detroit, MI, USA, October 2002. SAE.

[2] R. Tappe and D. Ehrhardt. Dynamic tests in complex systems. In *Proceedings of the International Test Conference*, pages 609–614. IEEE, 2001.

[3] M. Lorell, J. Lowell, M. Kennedy, and H.P. Levaux. *Cheaper, Faster, Better? Commercial Approaches to Weapons Acquisition.* RAND Corporation, 2000.

[4] H. Kopetz, R. Obermaisser, P. Peti, and N. Suri. From a federated to an integrated architecture for dependable embedded real-time systems. Technical Report 22, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2004.

[5] F. Cristian. Understanding fault-tolerant distributed systems. *Communications of the ACM*, 34(2):56–78, 1991.

[6] J.C. Laprie. *Dependability: Basic Concepts and Terminology*. Springer Verlag, Vienna, Austria, 1992.

[7] D. Powell. Failure mode assumptions and assumption coverage. In *Proceedings of the 22nd IEEE Annual International Symposium on Fault-Tolerant Computing (FTCS-22)*, pages 386–395, Boston, USA, July 1992.

[8] M. Hsueh, T.K. Tsai, and R.K. Iyer. Fault injection techniques and tools. *IEEE Computer*, 30(4):75–82, April 1997.

[9] C.J. Walter, P. Lincoln, and N. Suri. Formally verified online diagnosis. *IEEE Transactions on Software Engineering*, 23(11):684–721, November 1997.

[10] Aeronautical Radio, Inc., 2551 Riva Road, Annapolis, Maryland 21401. *ARINC Specification 624: Design Guidance for Onboard Maintenance System*, August 1993.

[11] J. Rushby. Partitioning for avionics architectures: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Also to be issued by the FAA.

[12] P. Peti, R. Obermaisser, and H. Kopetz. An integrated diagnostic architecture. Research report, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2004.

[13] R. Obermaisser, P. Peti, and H. Kopetz. Virtual networks in an integrated time-triggered architecture. Research report, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2004.

[14] J. Gait. A probe effect in concurrent programs. *Software Practice and Experience*, 16(3):225–233, March 1986.

[15] A. Avizienis, J.C. Laprie, and B. Randell. Fundamental concepts of dependability. Research Report 01-145, LAAS-CNRS, Toulouse, France, April 2001.

[16] B. Pauli, A. Meyna, and P. Heitmann. Reliability of electronic components and control units in motor vehicle applications. In *VDI Berichte 1415, Electronic Systems for Vehicles*, pages 1009–1024. Verein Deutscher Ingenieure, 1998.

[17] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[18] J.H. Lala and R.E. Harper. Architectural principles for safety-critical real-time applications. *Proceedings of the IEEE*, 82:25–40, January 1994.

[19] H. Kim, A.L. White, and K.G. Shin. Effects of electromagnetic interference on controller-computer upsets and system stability. *IEEE Transactions on Control Systems Technology*, 8(2):351–357, March 2000.

[20] J. Swingler, J.W. McBride, and C. Maul. Degradation of road tested automotive connectors. *IEEE Transactions on Components and Packaging Technologies*, 23(1):157–164, March 2000.

[21] N.E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Transactions on Software Engineering*, 26(8):797–814, August 2000.

[22] S. Mishra, M. Pecht, and D.L. Goodman. In-situ sensors for product reliability monitoring. In *Proceedings of SPIE*, volume 4755, pages 10–19, 2002.

[23] M. Pecht and V. Ramappan. Are components still the major problem: a review of electronic system and device field failure returns. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, 15(6):1160–1164, December 1992.

[24] C. Constantinescu. Impact of deep submicron technology on dependability of VLSI circuits. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 205–209. IEEE, 2002.

[25] L. Lev and P. Chao. Down to the wire. Technical report, Cadence Design Systems, Inc., San Jose, CA, USA, 2002.

[26] DoD. *Military Handbook, Electronic Reliability Design Handbook (MIL-HDBK-338)*. US Department of Defense, 1998.

[27] M. Pecht. Electronic reliability engineering in the 21st century. In *Proceedings of 2001 International Symposium on Electronic Materials and Packaging*, pages 1–7. IEEE, 2001.

[28] B. Pauli and A. Meyna. Ein praxisorientierter Ansatz zur Bestimmung von komulierten und durchschnittlichen Ausfallraten. *Automobiltechnische Zeitschrift (ATZ)*, pages 382–386, 1998.

[29] J. Wilde, W. Wondrak, and W. Senske. Reliability requirements for microtechnologies used in automotive applications. In *Proceedings of the Congress for Microsystems and Precision Engineering, MicroEngineering 99*, Stuttgart, Germany, October 1999. Stuttgarter Messe- und Kongressgesellschaft GmbH.

[30] *IEEE standard methodology for reliability prediction and assessment for electronic systems and equipment*, January 1999. IEEE Std 1413-1998.

[31] A. Ramakrishnan et. al. *The Avionics Handbook*, chapter Electronic Hardware Reliability. CRC Press LCC, 2001.

[32] J.M. Wetzer, G.J. Cliteur, W.R. Rutgers, and H.F.A. Verhaart. Diagnostic- and condition assessment-techniques for condition based maintenance. In *Proceedings of the 2000 Annual Report Conference on Electrical Insulation and Dielectric Phenomena*, volume 1, pages 47–51, 2000.

[33] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni. Discriminating fault rate and persistency to improve fault treatment. In *Proceedings of The Twenty-Seventh Annual International Symposium on Fault-Tolerant Computing (FTCS'97)*, pages 354–362, Washington - Brussels - Tokyo, June 1997. IEEE.

[34] G. Heiner and T. Thurner. Time-triggered architecture for safety-related distributed real-time systems in transportation systems. In *Proceedings of the Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing*, pages 402–407, June 1998.

[35] International Standardization Organisation, ISO 7637. *Road vehicles – Electrical disturbances from conduction and coupling*, 1995.

[36] A. Schedl. The short-term stability of crystal oscillators: Experimental results. Research Report 1/1995, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 1995.

[37] P. Gil, J. Arlat, H. Madeira, Y. Crouzet, T. Jarboui, K. Kanoun, T. Marteau, J. Duraes, M. Vieira, D. Gil, J.C. Baraza, and J. Gracia. *Fault Representativeness*. LAAS-CNRS, Toulouse, France, 2002. Deliverable (ETIE2) of the European Project Dependability Benchmarking Dbench (IST-2000-25425).

[38] W. Wondrak. Physical limits and lifetime limitations of semiconductor devices at high temperatures. *Microelectronics Reliability*, 39:1113–1120, 1999.

[39] D. Galler and G. Slenski. Causes of aircraft electrical failures. *Aerospace and Electronic Systems Magazine*, 6(8):3–8, August 1991.

[40] S. Sullivan and G. Slenski. Managing electrical connection systems and wire integrity on legacy aerospace vehicles. In *Proceedings of the FAA Principal Inspectors and Engineers Workshop*, Seattle, USA, 2001.

[41] US AirForce. Aircraft mishap investigation handbook for electronic hardware. Technical Report WL-TR-95-4004, Materials Directorate, Wright Laboratory, Air Force Material Command, January 1995.

[42] J.W. McBride. Electrical contact and connectors in automotive systems. In *Proceedings of the IEE Colloquium on Connectors on Vehicles*, pages 3/1–3/7, November 1993.

[43] P.N. Tanner. Automotive connectors. In *Proceedings of the IEE Colloquium on Connectors on Vehicles*, pages 6/1–6/10, November 1993.

[44] K. Kimseng, M. Hoit, N. Tiwari, and M. Pecht. Physics-of-failure assessment of a cruise control module. *Microelectronics Reliability*, 39:1423–1444, 1999.

[45] E. Normand. Single-event effects in avionics. *IEEE Transactions on Nuclear Science*, 43(2):461–474, April 1996.

[46] E. Normand. Single event upset at ground level. *IEEE Transactions on Nuclear Science*, 43(6):2742–2750, December 1996.

[47] A.S. Podgorski. Lightning standards for aircraft protection. In *Proceedings of the IEEE International Symposium on Electromagnetic Compatibility*, pages 218–223, August 1990.

[48] I.E. Noble. Electromagnetic compatibility in the automotive environment. *Science, Measurement and Technology*, 141(4):252–258, July 1994.

[49] I. Berger. Can you trust your car? *IEEE Spectrum*, 39(4):40–45, April 2002.

[50] I.E. Noble. Emc and the automotive industry. *Electronics & Communication Engineering Journal*, 4(5):263–271, October 1992.

[51] G. McCall, D. Newman, and D. Ward. Guidance on automotive software development in relationship to emc. In *Proceedings of the IEE Colloquium on Electomagnetic Compatibility of Software*, pages 8/1–8/5, November 1998.

[52] M.J. Pont, R. Kureemun, H.L.R. Ong, and W. Peasgood. Increasing the reliability of embedded automotive applications in the presence of emi: a pilot study. In *Proceedings of the IEE Seminar on Electromagnetic Compatibility for Automotive Electronics*, pages 4/1–4/4, September 1999.

[53] W.J. Fleming. Overview of automotive sensors. *IEEE Sensors Journal*, 1(4):296–308, December 2001.

[54] W. Wondrak, A. Boos, and R. Constapel. Design for reliability

in automotive electronics. In *Proceedings of the Microtec 2000*, Hannover, EXPO, July 2000.

[55] C. Furse and R. Haupt. Down to the wire [aircraft wiring]. *IEEE Spectrum*, 38(2):34–39, February 2001.

[56] J. Gray. Why do computers stop and what can be done about it? In *Proceedings of the 5th Symposium on Reliablity in Distributed Software and Database Systems*, January 1986.

[57] D. Wybo and D. Putti. A qualitative analysis of automatic code generation tools for automotive powertrain applications. In *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design*, pages 225–230, Hawaii, USA, August 1999.

[58] MISRA. *Guidelines for the use of the C language in vehicle based software*. The Motor Industry Software Reliability Association (MISRA), April 1998.

[59] J.P. Weber. Integrated diagnostics for software. In *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference*, volume 2, pages 565–571, May 1992.

[60] R.A. George and C.J. Wang. Vehicle E/E system integrity from concept to customer. In *Convergence International Congress & Exposition On Transportation Electronics*, Detroit, MI, USA, October 2002. SAE.

[61] E.N. Adams. Optimizing preventive service of software products. *IBM Journal of Research and Development*, 28(1):2–14, 1984.

[62] Robert Bosch GmbH, editor. *Autoelektrik Autoelektronik*. Vieweg Verlag, Braunschweig/Wiesbaden, 4th edition, 2002.

[63] I. Moir and A. Seabridge. *Civil Avionics Systems*. Professional Engineering Publishing, London, UK, 2003.

[64] M. Parsons. Design and manufacture of automotive pressure sensors. *Sensors*, April 2001.

[65] B. Peischl and F. Wotawa. Model-based diagnosis or reasoning from first principles. *IEEE Intelligent Systems*, 18(3):32–37, May 2003.

[66] M. Nyberg. Model-based diagnosis of an automotive engine using several types of fault models. *IEEE Transactions on Control Systems Technology*, 10(5):679–689, September 2002.

[67] S. Poledna. TTP-Tools – The tool set of the Time-Triggered Architecture. In *Proceedings of the Monterey Workshop*, Baden, Austria, October 2004.

[68] L. Kleinrock. *Queuing Systems Volume I: Theory*. John Wiley and Sons, New York, 1975.

[69] P. Peti, R. Obermaisser, and H. Kopetz. Out-of-norm assertions. Research Report 42/2004, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2004.

[70] H. Kopetz. Sparse time versus dense time in distributed real-time systems. In *Proceedings of 12th International Conference on Distributed Computing Systems*, Japan, June 1992.

[71] H. Kopetz and N. Suri. Compositional design of RT systems: A conceptual basis for specification of linking interfaces. In *Proceedings of Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 51–60, May 2003.

[72] S. Amberkar and B. Murray. Diagnostic strategies for advanced automotive systems. In *Convergence International Congress & Exposition On Transportation Electronics*, Detroit, MI, USA, October 2002. SAE.