RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

International Workshop on the
Design of Dependable Critical Systems

ECOMODIS

# Proceedings

## of the
## International Workshop on the
## Design of Dependable Critical Systems
## "Hardware, Software, and Human Factors in Dependable System Design"

# DDCS 2009

**September 15, 2009**
**Hamburg, Germany**

**In the framework of**
**The 28th International Conference on**
**Computer Safety, Reliability and Security**
**SAFECOMP 2009**

**Edited by**

**Achim Wagner[1], Meike Jipp[1], Colin Atkinson[2] and Essameddin Badreddin[1]**

**[1] Automation Laboratory, Institute of Computer Engineering, University of Heidelberg**
**[2] Chair of Software Engineering, University of Mannheim**

**ISBN 978-3-00-029877-6**

ziti

# Organization

## *Editors*

Achim Wagner[1], Meike Jipp[1], Colin Atkinson[2] and Essameddin Badreddin[1]
[1] Automation Laboratory, Institute of Computer Engineering, University of Heidelberg
[2] Chair of Software Engineering, University of Mannheim

## *Programme Committee*

Ciamak Abkai
(University of Heidelberg)
Colin Atkinson
(University of Mannheim)
Essameddin Badreddin
(University of Heidelberg)
Christian Bunse
(International University in Germany)
Frederic Diederichs
(Fraunhofer IAO, Germany)
Daniel Görlich
(DFKI, German Research Center for Artificial Intelligence)
Hans-Gerhard Gross
(Delft University of Technology)
Marcel Held
(EMPA, Swiss Federal Laboratories for Materials Testing and Research)
Meike Jipp
(University of Heidelberg)
Gerrit Meixner
(DFKI, German Research Center for Artificial Intelligence)

## *Organizing Committee*

Essameddin Badreddin
(University of Heidelberg)
Meike Jipp
(University of Heidelberg)
Achim Wagner
(University of Heidelberg)
Ciamak Abkai
(University of Heidelberg)

## *Contact*

Achim Wagner
University of Heidelberg
Automation Lab
B6, 26, Building B0.09
68131 Mannheim
Phone: +49 621181 3048
Email: ecomodis@ziti.uni-heidelberg.de

# Information on Publication

To ensure a high level of academic content, a peer review process has been used. Each submission has been reviewed by a minimum of two separate reviewers on the Programme Committee list.
The proceedings are available electronically at the website of HeiDOK, the Open Access document server of the University of Heidelberg (see links below). This publication platform offers free access to full-text documents and adheres to the principles of OpenAccess as well as the goals of the Budapest Open Access Initiative (BOAI). The papers are accessible through a special sub-portal and are fully citable.
The Open Access Document Server of the University library of Heidelberg also offers the possibility to order hardcopies of the proceedings.

Open Access Document Server:
http://archiv.ub.uni-heidelberg.de/volltextserver/portal/ddcs2009
Workshop Website:
http://www.ecomodis.de/cms/DDCS
Research group ECOMODIS:
http://www.ecomodis.de

# Workshop Scope

## *Abstract*

As technology advances, technical systems become increasingly complex not only in terms of functionality and structure but also regarding their handling and operation. In order to keep such complex safety-critical and mission-critical systems controllable, they are required to be highly dependable. Since the costs for designing, testing, operating, and maintaining such systems significantly increase with the dependability requirements, new design approaches for the cost effective development and production of dependable systems are required, covering hardware, software, and human factor aspects. This workshop aims at presenting and discussing the latest developments in this field, spanning the entire spectrum from theoretical works on system architecture and dependability measures to practical applications in safety and mission critical domains.

Topics of interest *include* but *are not restricted to* the following:

## *Applications*

- Medical and rehabilitation technology
- Assistance systems
- Automotive and aerospace
- (Semi-) Autonomous control systems
- Robotics

## *Research Areas*

- Dependability analysis and modelling for Hardware, Software, and Human Factors
- System architectures
- Component-based design of Hardware/Software/Human Factors
- Monitoring and testing
- Fault-tolerant control and dependable system reconfiguration

For further information visit the workshop homepage
http://www.ecomodis.de/cms/DDCS

# Workshop Programme

## 15 September 2009, Hamburg, Germany

**Hamburg University of Applied Sciences**
**Berliner Tor, Building A**

*Oral presentation session (9:00-12:45)*

09:00 – 09:25 **Observation-Based Modeling for Testing Highly Dependable Systems – A Practitioner's Approach**

Teemu Kanstrén, Èric Piel, Alberto Gonzalez, and Hans-Gerhard Gross

09:25 – 09:50- **Safety Recommendations for Safety-Critical Design Patterns**

Ashraf Armoush and Stefan Kowalewski

09:50 – 10:15 **Towards a Practical, Unified Dependability Measure for Dynamic Systems**

Achim Wagner, Colin Atkinson and Essameddin Badreddin

10:15 – 10:40 **Measuring the Dependability of Dynamic Systems using Test Sheets**

Colin Atkinson, Florian Barth and Giovanni Falcone

10:40- 11.05 **Coffee break**

11:05 – 11:30 **Fault Propagation Analysis on the Transaction-Level Model of an Acquisition System with Bus Fallback Modes**

Raul S. Fajardo Silva, Jürgen Hesser, Reinhard Männer

11:30 – 11:55 **The Impact of Individual Differences in Fine Motor Abilities on Wheelchair Control Behavior and Especially on Safety-Critical Collisions with Objects in the Surroundings**

Meike Jipp, Christian Bartolein, Achim Wagner, Essameddin Badreddin

11:55 – 12:20 **Real-Time Physiological Simulation and Modeling toward Dependable Patient Monitoring Systems**

Ciamak Abkai, Jürgen Hesser

12:20 – 12:45 **An Integrated Monitor-Diagnosis-Reconfiguration Scheme for (Semi-) Autonomous Systems**

Yi Luo, Achim Wagner, Essameddin Badreddin

12:45 – 14:00 **Lunch**

*Poster and demonstration session (14:00-15:20)*

*Posters*

1. **Quantifying Safety in Software Architectural Designs**
   Atef Mohamed, Mohammad Zulkernine
2. **The Role of Task and Situational Characteristics for the Dependability of Human-Technology Interaction**
   Meike Jipp, Christian Bartolein, Essameddin Badreddin
3. **Hierarchical Hybrid Monitoring for Autonomous Systems**
   Leila Zouaghi, Achim Wagner, Essameddin Badreddin
4. **Dependable Design for Assistance Systems: Electrical Powered Wheelchairs**
   Christian Bartolein, Achim Wagner, Essameddin Badreddin
5. **System Testing using Test Sheets**
   Colin Atkinson, Florian Barth, Daniel Brenner

*Demonstrations:*

**Dependable component-based design on the Example of a Heating Control System**
Leila Zouaghi, Markus Koslowski, Alexander Alexopoulos

15.20 – 15.45 **Coffee break**

*Round table discussion (15:45-17:00)*

# Table of Contents

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# Observation-Based Modeling for Testing and Verifying Highly Dependable Systems – A Practitioner's Approach

Teemu Kanstrén[1], Eric Piel[2], Alberto Gonzalez[2], and Hans-Gerhard Gross[2]

[1] VTT, Kaitoväylä 1, Oulu, Finland
teemu.kanstren@vtt.fi
[2] Delft University of Technology, Mekelweg 4, 2628 CD Delft
{e.a.b.piel,a.gonzalezsanchez,h.g.gross}@tudelft.nl

**Abstract.** Model-based testing (MBT) can reduce the cost of making test cases for critical applications significantly. Depending on the formality of the models, they can also be used for verification. Once the models are available model-based test case generation and verification can be seen as "push-button solutions." However, making the models is often perceived by practitioners as being extremely difficult, error prone, and overall daunting.
This paper outlines an approach for generating models out of observations gathered while a system is operating. After refining the models with moderate effort, they can be used for verification and test case generation. The approach is illustrated with a concrete system from the safety and security domain.

## 1 Introduction

Testing consumes a large portion of the overall development cost for a software project. Because testing adds nothing in terms of functionality to the software, there is a strong incentive towards test automation with Model-Based Testing (MBT). Once the models are made and appropriate tools are available, MBT is a push-button solution. Making the models of the System Under Test (SUT), to be used for automated processing and test case generation, does not add any immediate auxiliary value to the final product as well. Moreover, it is typically perceived by practitioners as being difficult, expensive, and overall daunting. One solution for circumventing the difficult and costly manual design and construction process to obtain models for MBT is to generate them out of observations automatically [5], e.g., with the aid of a process mining technique [9].

Obviously, this method of observation-based modeling has to be "bootstrapped" and, therefore, works only on existing software with existing runtime scenarios, e.g., field data and existing test suites [2]. Because most typical software projects in practice have test suites, Observation-Based Modeling (OBM) can be adopted easily by practitioners, and can, eventually, offer automated support for constructing system specification models to be used for system testing following system evolution.

This article presents and outlines a method for model-based testing driven by observation-based modeling. The method is supported by a compilation of existing techniques and tools that have been combined and integrated in order to devise a practical, iterative and (semi-) automatic way to support the creation of behavioural models out of execution traces (observations). The models are made specifically for model-based testing, and they are applied to test and verify a component of a maritime safety and security system. Evaluation of the proposed approach indicates that system specification models for a security system can be boot-strapped from existing execution scenarios, and that they can be refined into models suitable for MBT with relatively little manual user involvement.

The paper is structured as follows. Sect. 2 presents work related, Sect. 3 describes our proposed approach of model-generation, verification, refinement, and testing. Sect. 4 presents evaluation of the work, and finally, Sect. 5 summarizes and concludes the paper with future directions.

## 2 Background and Related Work

OBM demands that (test) executions of the system under test can be observed, also referred to as tracing. Tracing is widely used in dynamic analysis of programs and it can be applied to observe which components, methods, or basic building blocks are invoked during execution, in order to turn this information into a behavioural model of the software [2]. In addition, external tracing mechanisms such as aspects [6] provide the advantage that the source code does not have to be amended for supporting the tracing.

Finite State Machines (FSM) and Extended FSM (EFSM) are of particular interest for behavioural modeling and, consequently, for behavioural testing [8]. They describe the system in terms of control states and transitions between those states. EFSM add guard conditions to the more general FSM.

Bertolino et al. [1] proposed three steps to the automated "reverse-engineering" of models to be used for model-based testing, but they never realized their proposition. Our method outlined here takes their ideas further and discusses a concrete implementation with existing tools. Ducasse et al. [4] use queries on execution traces to test a system. In this article, we apply similar techniques to help understand what a system does, and to test it. D'Amorim et al. [3] apply symbolic execution and random sequence generation for identifying method invocation sequences of a running system. They devise the test oracle from exceptions and from monitoring executions violating the system's operational profile, described through an invariant model. Our proposed method follows their approach of generating the test oracle. Lorenzoli et al. [7] present a way to generate EFSM from execution traces, based on FSM and Daikon[3]. They use the EFSM for test case selection in order to build an optimal test suite.

[3] `http://groups.csail.mit.edu/pag/daikon`

## 3 Observation-Based Modeling

Observation-Based Modeling turns the traditional MBT approach around as described in [1]. Instead of creating a model manually, based on a (non-formal) specification, the model is created from the implementation, based on executing a limited number of initial test cases, and tracing their executions. OBM can be used to generate the test model for MBT, the test harness, and the test oracle, by monitoring the SUT's input and output during a selected set of execution scenarios. The entire process can be divided in four different activities, as detailed below.

### 3.1 Capturing a set of observations

The first step in OBM is to capture a suitable set of observations to be used as a basis for the initial model generation. To obtain observations, the SUT behaviour is monitored while exercising it using a set of existing execution scenarios, such as existing test cases, recorded user sessions, or field data [2].

The main information required to be captured are the *messages passed* through the input- and output-interfaces of the SUT, and the *SUT internal state* each time a message is passed. Typical component middlewares allow to list the component interfaces and to capture all component interactions, without having to instrument every component individually. Obtaining the internal state might be harder, as our approach strives to be compatible with black-box components. Accessing this information typically requires an additional test interface or serialization interface designed into the SUT. In case this is lacking, either the SUT must be manually extended, or it could be possible to maintain an "artificial" state out of the inputs and outputs.

### 3.2 Automatic generation of the model

The second activity consists in processing those traces and generating an initial behavioural model. This model, expressed as an EFSM, requires the production of states, transitions, and guards.

The generation of the initial EFSM comprises four phases. First, the static parts of the model are generated. These parts are similar for all generated models, and the provided SUT interface definitions are the variables used as input in this phase. Second, an FSM is generated which describes the SUT in terms of interface usage, where each message passed through one of the interfaces matches a state in the FSM. This is done via the ProM tool [9]. This FSM is analysed and processed with specific algorithms to capture the interactions (states and transitions) for the EFSM. Third, invariants over the SUT internal state and parameter data values are provided, and then used to generate constraints, i.e., transition guards, for the interactions and for the processed data values (input data). Finally, all these separate parts of the model are combined to produce the complete EFSM. Fig. 1 presents a very simple example of EFSM specified in the same way as a model generated by our tool. The current state of the model

is reported by one special method `getState()`. Every transition is described by one method (e.g.: `vend()`) plus an associated method describing its guard (e.g.: `vendGuard()`).

```
public class VendingMachineModel implements FsmModel {
  private int money = 0;

  public Object getState() { return money; }
  public void reset(boolean b) { money = 0; }

  @Action public void vend() {money = 0;}
  public boolean vendGuard() {return money == 100;}

  @Action public void coin25() {money += 25;}
  public boolean coin25Guard() {return money <= 75;}

  @Action public void coin50() {money += 50;}
  public boolean coin50Guard() {return money <= 50;}
}
```

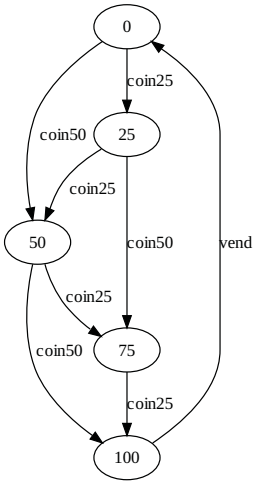



**Fig. 1.** Example EFSM of a vending machine.

### 3.3 Test execution

In order to generate the test cases out of the EFSM, our approach relies on ModelJUnit[4]. A test case is created for every possible path going through the various states, along the transitions. Let us note that in this type of model, the lack of some states or transitions compared to the "perfect" model signifies only that the modeled behaviour is not complete, but still represents only allowed behaviour. It is therefore possible to run the test execution before the model is finalized.

In our approach, each transitions contains code to actually send and listen messages from the SUT. Each transition also contains JUnit[5] assertions to determine if the correct messages were answered. The triggering of an assertion implies failure of the test case. The test case is considered passed if no assertion was triggered during the entire execution of the path.

### 3.4 Manual refinement

The fourth activity for the MBT consists in manual improvement of the generated EFSM. It is typically performed *in parallel* to the test execution activity.

[4] `http://czt.sourceforge.net/modeljunit`
[5] `http://www.junit.org`

In addition to defining the initialization of the complex variables, the task of the engineer is to refine and generalize the EFSM to match the generic expected behaviour of the SUT, which might be different from the observed behaviour. This manual activity should be done in little gradual steps, guided by the results of the tests which exercise the new paths introduced by the generalization of the model at the precedent step.

## 4 Case study

An example SUT called Merger is used to illustrate the techniques. It is part of a maritime surveillance system. It receives information broadcasts from ships called AIS messages and processes them in order to form a situational picture of the coastal waters. The Merger acts as temporary database for AIS messages, and client components can consult it for track information of ships, or receive notifications of certain ship events. The SUT is also used by software controlling the main screen in the command and control centre for displaying ship tracks. The system comes with a specification in plain English defining behaviour and communication protocols of its components. The components are implemented in Java, executed under the Fractal component framework[6].

### 4.1 Qualitative evaluation

The Merger component was first instrumented to allow observing a few variables representing its global state and the method calls. Then, observation of the component was performed while 6 manually written unit tests were run and during five minutes of normal operation with field input data. An EFSM was generated out of the traces. This model was manually refined (470 lines of code were changed over 1700) mainly by defining initialization code, generalizing the guards, and correcting the expected behaviour depending on the specification. The refinement process took place with feedback from the generated tests which gradually tested more behaviour of the component. When a bug in the Merger component was found, it was immediately fixed and the refinement process resumed. The refinement and testing process was finished when all the states were accessed, and none of the generated test cases failed.

During this process several errors were found. These errors can be classified into three main types: mismatches between implementation and specification (3), ambiguities in the specification (1), and problems in the design that cause errors under certain conditions specific to the testing environment (2). Overall, in terms of identifying previously unknown errors of a component that had been used for some time in this context, this can be regarded as a very successful model-based testing experiment with real value to the quality of the system.

The evaluation of the method presented, performed in the Merger case study, indicates that the models generated can be used well for model-based testing after moderate manual amendments.

[6] `http://fractal.ow2.org`

### 4.2 Quantitative evaluation

In order to evaluate the efficiency of the approach, two quantitative evaluations have been also performed. First, using the source code of the Merger component, the coverage of the generated tests has been measured. The measurements are shown in Table 1. Here, *Unit tests* refer to the six initial unit tests used as execution scenarios. *EFSM* refers to the tests generated by the MBT tool out of the final refined model. The four columns correspond to four different types of coverage: statement, method, conditional, and path coverage. This latter one is the number of unique paths in the final EFSM which were followed during a test.

| Source | Statements | Methods | Conditionals | Paths |
|---|---|---|---|---|
| Unit tests | 53.5% | 64.5% | 38.7% | 6 |
| EFSM | 64.1% | 67.7% | 48.4% | 87 |
| EFSM + Unit tests | 65.5% | 67.7% | 51.6% | 92 |

**Table 1.** SUT coverage breakdown by execution scenarios.

It is visible that the tests generated from the model provide a significant increase in coverage over the used unit tests. The EFSM set outperformed the initial tests by a small percentage due to observation of the system also on field data, as well as due to the generalization of the generated model in the verification and testing refinement phase. This generalization permitted execution of additional parts of the code, while most parts executed by the original tests are still executed. The biggest difference is in the Paths metric. EFSM largely outperforms the initial tests. Nevertheless this is what is to be expected from an MBT tool that is intended to generate complex interactions to test the SUT.

Second, mutation testing was used to evaluate the effectiveness of the generated test suite. Mutation testing consists in introducing a modification in the code of SUT, and to check whether a test suite is able to detect this "mutation". 117 "mutants" were automatically generated, of which 51 were considered semantically equivalent after manual inspection. The results are shown in Table 2. When a test finds no errors (the SUT is considered to operate fine), the result is termed "positive", and oppositely, when an error is reported, it is termed "negative". "False positives" are the mutations reported fine although it was manually verified that they behave outside of the specification sometimes. "False negative" would be a case where a correct SUT is classified as having an error.

| Source | True positive | False positive | True negative | False negative | Total |
|---|---|---|---|---|---|
| Unit tests | 51 | 16 | 50 | 0 | 117 |
| EFSM | 51 | 15 | 51 | 0 | 117 |

**Table 2.** Mutation test results.

The final model provides minimal gain over the initial unit tests. The model outperforms the unit tests in correct categorization of mutants with actual modified behaviour only by a slight margin. Nevertheless, it is worthy to note that the

correct categorizations done by the EFSM are a superset of the one performed by the unit tests. The generated model could detect all the bugs originally detected by the units tests and more.

## 5 Conclusions

Dependable systems need a high quality of engineering in order to ensure the stability and the correct behaviour of the implementation. Models are useful assets for system engineering. They can be used for specification, verification, reasoning, and test case generation. Once models are available, powerful tools and techniques can be applied to support a range of activities. However, making the models is still perceived by practitioners as being difficult, costly, and error prone. A way to circumvent the difficult modeling process is to have specification models derived automatically from observations from a running system. Because such models specify observed behaviour, rather than expected behaviour, they have to be amended, in order to be applied, eventually, for verification and test case generation.

This paper has presented a approach to bootstrap, refine, and verify models from execution traces, to be used primarily for model-based testing.

## References

1. A. Bertolino, A. Polini, P. Inverardi, and H. Muccini. Towards anti-model-based testing. In Fast Abstract in The Int'l. Conf. on Dependable Systems and Networks, DSN 2004, Florence, 2004.
2. B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke. A systematic survey of program comprehension through dynamic analysis. IEEE Transactions on Software Engineering, 2009.
3. M. d'Amorim, C. Pacheco, D. Marinov, T. Xie, and M.D. Ernst. An emprical comparison of automated generation and classification techniques for object-oriented unit testing. In: 21st Intl. Conf. on Automated Software Engineering (ASE'06), pp. 59–68, Tokyo, Japan, Sept. 2006.
4. S. Ducasse, T. Girba, and R. Wuyts. Object-oriented legacy system trace-based logic testing. In: Conf. on Software Maintenance and Reengineering (CSME'2006), pp. 37–46, 2006.
5. A. Hamou-Lhadj, E. Braun, D. Amyot, and T. Lethbridge. Recovering behavioral design models from execution traces. In 9th European Conf. on Software Maintenance and Reengineering (CSMR'2005), pages 112–121, 2005.
6. G. Kiczales, E. Hilsdale, J. Hugumin, M. Kersten, J. Palm, and W. Griswol. Getting started with AspectJ. Communcation of the ACM, 44(10):59–65, 2001.
7. D. Lorenzoli, L. Mariani, and M. Pezze. Automatic generation of software behavioral models. In 30th Intl. Conf. on Software Engineering (ICSE'08), pp. 501–510, Leipzig, 2008.
8. M. Uttig and B. Legeard. Practical Model-Based Testing: A Tools Approach. Morgan Kaufman, 2006.

9. W. M. P. van der Aalst, B. F. van Dongen, C.W. Gnther, R. S. Mans, A. K. A. de Medeiros, A.Rozinat, V. Rubin, M.Song, H. M. W. Verbeek, and A. J. M. M. Weijters. Prom 4.0: Comprehensive support for real process analysis. In Application and Theory of Petri nets and Other Models of Concurrency 2007, volume 4546, pages 484–494. Springer, Berlin, Germany, 2007.

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# Safety Recommendations for Safety-Critical Design Patterns

Ashraf Armoush and Stefan Kowalewski

Embedded Software Laboratory
RWTH Aachen University
Aachen, Germany
{armoush, kowalewski}@embedded.rwth-aachen.de

**Abstract.** The concept of design patterns, which is considered as one of the commonly used techniques in the development of software and hardware systems, is applicable to be used in the design of safety-critical embedded systems. While several safety metrics and assessment methods have been proposed to evaluate safety-critical systems, most of these methods cannot be used for safety-critical design patterns, due to the fact that a design pattern presents a high-level abstract solution to commonly recurring design problem and it is not related to a specific application or to a specific case. This paper proposes a system of safety recommendations for safety-critical design patterns, which can be used in the assessment of design patterns for safety-critical embedded systems to reflect the severity of failure in the target application. The proposed safety recommendations are based on the safety recommendations of the IEC 61508 standard, and contain additional 3 types of recommendations: weakly not recommend, weakly recommended, and moderately recommended.

**Key words:** Safety-Critical, Design Patterns, Safety Recommendations

## 1 Introduction

With the increasing use of the concept of design patterns as a universal approach to describe common solutions to widely recurring design problems in software and hardware domain, it has become a good candidate for the design of safety-critical embedded systems. In general, the safety-critical systems address the applications in which failure can lead to serious injury, significant property damage, or damage to the environment [1, 2]. The design of these systems should fulfill the intended functional requirements (FR) as well as non-functional requirements that define qualities of a system such as: safety, reliability, and execution time.

System safety represents the main non-functional requirement for safety-critical embedded system. It is defined by *MIL-STD-882D* standard [3] as "Freedom from those conditions that can cause death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment".

Therefore, many design techniques, concepts, safety methods, metrics and standards have been proposed and used to cover the development lifecycle and to improve the non-functional requirements of such safety-critical systems.

While the commonly used standards give recommendations for safety design techniques, none of these standards gives recommendations at the level of design patterns. In most situations, design patterns present abstract solutions that combine more than single safety technique and design method to solve some common design problems. Therefore, a system of recommendations should be used for safety-critical design patterns to facilitate the comparison process and safety assessment for these patterns.

In this paper, we propose a systematic method to find safety recommendations for safety-critical design patterns. This method is based on the recommendations proposed by *IEC 61508* standard [4] for safety techniques. The proposed safety recommendations, which reflect the importance of the addressed design pattern and severity of target applications, can be used later as a part of a safety assessment method for design patterns.

## 2 The Concept of Design Patterns

The idea of design patterns was original proposed by the architect "*Christopher Alexander*" [5], then it became one of the widely used techniques to support designers and system architects in their choice of suitable solutions for commonly recurring design problems.

While this concept has been applied in several application domains of hardware and software design (see e.g. [6][7]), further research is still needed to adapt this concept for the field of safety-critical embedded systems. The design of these systems is considered to be a complex process, since there are many non-functional requirements, mainly safety, that have to be fulfilled by these systems to assure that the risk of hazards is acceptable low.

Due to the fact that the current representations of design patterns lack a consideration of potential side effects on non-functional requirements, we proposed in a previous paper [8] a pattern representation template for safety-critical embedded application design methods. This representation includes the traditional pattern concept in combination with an extension describing the implications and side effects of the represented design method on the non-functional requirements of the overall system.

In order to facilitate the safety comparison process between the design patterns under consideration, a system of safety recommendations for safety-critical design patterns should be constructed. Later, these recommendations can be used in a safety assessment method for safety-critical design patterns.

## 3 Safety Standards and Risk Metric

In the design of safety-critical embedded systems, specific aspects, requirements, techniques, and safety management have to be considered. Thus, many safety

standards have been proposed to cover the safety management of safety-critical systems throughout their lifecycles. Some of the important and commonly used standards are: *MIL-STD-882D* [3] which is a military standard, and *IEC 61508* [4] which is a well-known application-independent standard.

Most of the time, system safety is related to the risk of failure in a system and the techniques that should be used to reduce the risk to an acceptable level. The safety assessment of safety-critical systems requires the use of a specific safety and risk metric. Thus, many metrics, such as *Steady-State Safety* ($S_{SS}$) and *Mean Time to Unsafe Failure* ($MTTUF$) (see e.g. [9][10]), have been proposed to be used in the assessment of safety-critical systems. Nevertheless, the risk, which is defined in the standard *IEC 61508* as a combination of the probability of occurrence of harm[1] and the severity of that harm, is considered as the most generic metric that deals with a wide range of applications. The risk metric is based on the following equation:

$$R = C \times f \tag{1}$$

– $R$: *the risk in the system.*
– $C$: *the consequence of the hazardous event*[2].
– $f$: *the frequency of the hazardous event.*

The aim of our research is to construct a catalog of safety-critical design patterns, and to develop a safety metric similar to the metric in Equation 1. This metric will be used as a safety and risk metric for the assessments of these patterns and to give an indication about the implication and side-effects of safety-critical design patterns on system safety.

### 3.1 Limitations:

A design pattern represents a high level abstract solution to a commonly recurring design problem. *It is not related to a specific application or to a specific case*; thus, it is very difficult to find an actual value for both the frequency of the hazardous event ($f$) and the consequence of the hazardous event ($C$).

In order to find parameters for our metric similar to the parameters used in the original risk metric shown in Equation 1, we have proposed in a previous paper [11] a metric that reflects the idea of frequency of hazardous event.(*see Section 5*). Furthermore, to cover the other part, we propose in the next section a method that gives an indication about the severity of the hazardous event similar to the first factor.

## 4 Applicability to Safety Integrity Levels

The *IEC61508*, which is the most commonly-used standard in industrial applications, defined the term *safety integrity* as “probability of a safety-related system

[1] Harm: physical injury or damage to the health of people either directly or indirectly as a result of damage to property or to the environment [4].
[2] Hazardous Event: a situation which results in harm [4].

satisfactorily performing the required safety functions under all the stated conditions within a stated period of time". Part 4 of the standard defines 4 discrete levels for specifying the safety integrity requirements of the safety functions to be allocated to the system. These levels range from safety integrity level 1 ($SIL1$) to safety integrity level 4 ($SIL4$) where safety integrity level 4 denotes the highest level of safety integrity.

The *IEC61508* standard introduces recommendations for techniques and measures to be applied in order to avoid safety-critial failures caused by hardware or software. These recommendations are dependent on the safety integrity levels and are classified according to their importance into 4 types:

* **HR**: The technique is highly recommended for this safety integrity level.
* **R**: The technique is recommended for this safety integrity level
* –: The technique has no recommendation for or against used.
* **NR**: The technique is positively not recommended for this safety integrity level.

You have to keep in mind that the use of these recommended techniques does not give any guarantee that the final design will satisfy the required safety integrity level. However, in order to get a certificate for a specific safety integrity level, the standard should be used in the complete lifecycle of the design process.

While a safety integrity level is derived from an assessment of risk, it is not a measure of risk [12]. The safety integrity level will be used in the first component of our approach as an indication of severity of failures in the considered application after using a specific design pattern. The applications, that require high safety integrity levels, include higher severity than the applications with lower integrity levels. Thus, the intended safety integrity level for a system can be used as an indication of the possible severity and consequence of a hazardous event in that system.

Though the IEC61508 introduces recommendations for techniques and measures to avoid safety-critical failures caused by hardware or software, these recommendations are derived from safety integrity levels and given for different design techniques but not for general design patterns or architectures. Normally, a design pattern combines more than one architecture technique to improve the system safety. Therefore, we introduce a systematic method to give general safety recommendations at the level of design patterns.

| Recommendations | Value |
|---|---|
| NR | 0 |
| --- | 1 |
| R | 2 |
| HR | 3 |

**Table 1.** Safety recommendations in *IEC 61508*

To find the recommendations of importance for a specific pattern for the safety integrity levels, integer equivalent values are assigned to the recommendations as shown in Table 1. Then the average value of the existing techniques is calculated for each safety integrity level.

The resulting average value may range between two integer numbers which makes the selection of the suitable recommendation more difficult. To solve this problem, we introduce a new system of recommendations for design patterns. These recommendations contain additional 3 types: weakly not recommend, weakly recommended, and moderately recommended. The new recommendation types are classified based on the average value as shown in Table 2.

| Recommendations | | Average Value (*Avg*) |
|---|---|---|
| **NR** | Not Recommended | $Avg \leq 0.4$ |
| **WNR** | Weakly Not Recommended | $0.4 \leq Avg \leq 0.6$ |
| -- | No Recommendation | $0.6 < Avg < 1.4$ |
| **WR** | Weakly Recommended | $1.4 \leq Avg \leq 1.6$ |
| **R** | Recommended | $1.6 < Avg < 2.4$ |
| **MR** | Moderately Recommended | $2.4 \leq Avg \leq 2.6$ |
| **HR** | Highly Recommended | $Avg > 2.6$ |

**Table 2.** Proposed system of recommendations for design patterns

### 4.1 Example

The *Safety Executive Pattern (SEP)* [13], which is a large scale pattern used to provide a centralized and consistent method for monitoring and controlling the execution of a complex procedure in case of failures, includes the following design techniques: *program sequence monitoring by a watchdog*, *test by redundant hardware*, *safety bag techniques*, and *graceful degradation*. According to the *IEC 61508* standard, the recommendations for these techniques are shown Table 3.

| *Techniques* | *SIL1* | *SIL2* | *SIL3* | *SIL4* |
|---|---|---|---|---|
| Program sequence monitoring (WD) | HR | HR | HR | HR |
| Test by redundant hardware | R | R | R | R |
| Safety bag techniques | --- | R | R | R |
| Graceful degradation | R | R | HR | HR |

**Table 3.** Safety recommendations of the used techniques in *SEP*

Conforming to Table 3, the average recommendations, which show the applicability of the safety executive pattern to be used for different safety integrity

levels, are calculated and demonstrated in Table 4. As shown in the example,

| *Pattern* | *SIL1* | *SIL2* | *SIL3* | *SIL4* |
|---|---|---|---|---|
| Safety executive pattern | R | R | MR | MR |

**Table 4.** Safety recommendations of *Safety Executive Pattern*

our approach can be used to provide an indication of the severity of failures in the required application that will use a specific design pattern, by establishing of the intended safety integrity level and by finding the recommendation for this design pattern for this safety integrity level.

## 5 Probability of unsafe failure

As shown previously, it is difficult to find actual values for the risk metric in design patterns since these patterns describe general abstract solutions. This part provides a brief description for the parameter that has been used in our approach to cover the second factor in the risk metric. In general, the main goal of safety design methods is to reduce the probability of unsafe failure in the considered system. Therefore, we will use the probability of unsafe failure ($P_{UF}$) as a part of our metric for the risk assessment. This probability will be calculated in our approach relative to the probability of unsafe failure in a *basic system* that includes a single design channel and does not include any specific safety function.

In a previous paper [11] we have proposed a metric called ***Relative Safety Improvement (RSI)***. This metric is defined as *"the percentage improvement in safety (reduction in probability of unsafe failure) relative to the maximum possible improvement which can be achieved when the probability of unsafe failure is reduced to* 0*"*.

For any design pattern, the relative safety improvement can be calculated as shown in Equation 2:

$$\begin{aligned} RSI &= \frac{P_{UF(new)} - P_{UF(old)}}{0 - P_{UF(old)}} \times 100\% \\ RSI &= (1 - \frac{P_{UF(new)}}{P_{UF(old)}}) \times 100\% \end{aligned} \tag{2}$$

- $RSI$: Relative Safety Improvement.
- $P_{UF(old)}$:Probability of unsafe failure in the basic system.
- $P_{UF(new)}$:Probability of unsafe failure in the design pattern.

This part of our metric can be used in the proposed approach: either through employment of a mathematical modeling for design patterns or by using simulation techniques to demonstrate the safety improvement in each design pattern.

## 6 Conclusion

Safety assessment in safety-critical systems design is considered as an essential step to ensure that the final system is safe. Thus, several assessment methods and standards have been proposed to cover this process. While these methods and standards present some risk metrics, none of them can be applied to safety-critical patterns that address abstract solutions to common problems instead of real systems.

In this paper, we propose a systematic method to find safety recommendations at the abstract level of safety-critical design patterns. This method is based on the recommendations proposed by *IEC 61508* standard for safety techniques. The proposed safety recommendations, which reflect the importance of the addressed design pattern and severity of target applications, can be used together with the previously proposed metric (*RSI*) as a safety assessment method for design patterns. The combination of these two parts covers the main parameters: the frequency of the hazardous events and the consequence of these events, given in the original risk metric.

While the proposed approach can be used to facilitate the comparison process between the design patterns under consideration, there are many other factors that should be taken into consideration to achieve a comprehensive comparison. These factors include, among others: reliability, costs, time overhead and maintainability.

## Acknowledgment

This work was supported by the German Academic Exchange Service (DAAD) under the program: *Research Grants for Doctoral Candidates and Young Academics and Scientists.*

## References

1. Knight, J.C.: Safety critical systems: challenges and directions. In: ICSE '02: Proceedings of the 24th International Conference on Software Engineering, New York, NY, USA, ACM (2002) 547–550
2. Dunn, W.R.: Designing safety-critical computer systems. Computer **36**(11) (2003) 40–46
3. : MIL-STD 882D–Standard Practice for System Safety. US Department of Defense (DOD) (2000)
4. IEC61508: Functional safety for electrical / electronic / programmable electronic safety-related systems. International Electrotechnical Comission (1998)
5. Alexander, C.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)
6. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-oriented software architecture: a system of patterns. John Wiley & Sons, Inc. New York (1996)

7. Gama, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Element of Reusable Object-Oriented Software. Addison-Wesley, New York (1997)
8. Armoush, A., Salewski, F., Kowalewski, S.: Effective pattern representation for safety critical embedded systems. In: 2008 International Conference on Computer Science and Software Engineering. Volume 4., IEEE CS (Dec. 2008) 91–97
9. DeLong, T., Smith, D., Johnson, B.: Dependability metrics to assess safety-critical systems. IEEE Transactions on Reliability **54**(3) (2005) 498–505
10. Yu, Y., Johnson, B.: The quantitative safety assessment for safety-critical software. In: Proc. 29th Annual IEEE/NASA Software Engineering Workshop. (2005) 150–162
11. Armoush, A., Beckschulze, E., Kowalewski, S.: Safety assessment of design patterns for safety-critical embedded systems. In: 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2009), IEEE CS (Aug. 2009)
12. Redmill, F.: Iec 61508: Principles and use in the management of safety. IEE Computing and Control Engineering **9**(10) (1998) 205–213
13. Douglass, B.: Real-Time Design Patterns. Addison-Wesley, New York (2003)

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# Towards a Practical, Unified Dependability Measure for Dynamic Systems

Achim Wagner[2], Colin Atkinson[1], and Essam Badreddin[2]

[1] Lehrstuhl für Softwaretechnik, Universität Mannheim,
68131 Mannheim, Germany
atkinson@informatik.uni-mannheim.de

[2] Automation Laboratory, University of Heidelberg
68131 Mannheim, Germany
{achim.wagner, baddredin@ziti.uni-heidelberg.de}

**Abstract.** Providing a practical measure of the dependability of dynamic systems, including software systems and components, has been an elusive goal of systems engineers for some time. Measures for static, individual, dependability-relevant properties (e.g. reliability, safety, availability etc.) are well understood, but to date there is no general and widely accepted way of combining these into a single dependability measure that can be used to assess a dynamic system's capability for specific applications. In this paper we present a practical approach for obtaining an Integrated Dependability Measure (IDM) by placing the onus on system developers and users to capture the acceptability of different behavior in the form of acceptability functions rather than by defining (or attempting to define) general purpose combinations of separately determined dependability ingredients (e.g. reliability, safety etc.).

**Keywords:** Dependability Measure, Dynamic Systems, Behaviour-based System description

## 1 Introduction

Dependability is a complex concept which attempts to measure the degree to which a user can rely on a system to provide a certain level of service in a certain context [1]. There is general consensus on the various ingredients that contribute towards the dependability of a system such as reliability, safety, availability etc., and these ingredients are well understood [2]. However, even for traditional static systems which exhibit only "hard wired" patterns of behavior as they execute, there is no accepted generic approach for combining these separate ingredients into a single, overall dependability measure, and for dynamic systems which change their state over time it is even less clear how a the overall dependability can be represented by a combination of these attributes. Most of the approaches are related to binary fault models such as fault trees, Markov models or Petri-Nets [3]. These models are functional abstractions of the real system and their coincidence with the real system's behavior is difficult to prove [4]. Since dependability in general can also be defined as the capability of a system to successfully and safely fulfill its mission [5], the purpose of the system must be taken into account explicitly within a dependability measure.

One basic problem with trying to define a single unified measure of dependability from the traditional ingredients (e.g. reliability, safety etc.) is that their combination is highly application specific [6]. Thus, for systems which must satisfy strict safety requirements, safety measures must be given a much higher weighting than other ingredients such as reliability. In contrast, for systems which must satisfy stringent reliability requirements, reliability measures must be given a much higher weighting than other dependability ingredients such as safety and availability etc. The net effect of dependability's sensitivity to application specific requirements is that it is effectively impossible to define a single, generic way of combining the individual measures into a single measure. The definition of single, unified measure has therefore remained elusive.

In this paper we present a practical approach for getting around this problem that switches the onus for combining the dependability ingredients to system developers and users on a case-by-case basis rather than on researchers to find a single generic combination approaches. This is achieved by requiring developers to extend the specification of system behavior with so called "acceptability functions". The approach is based on a behavioral description of the system and the measurement and assessment of the system outputs [5]. In contrast with traditional specifications which merely describe the expected behavior of the system in response to stimuli, a specification enhanced with acceptability functions describes "how well" the range of possible behaviors of the system meet the requirements. In other words, an acceptability function describes (in terms of a value between (0..1) how "acceptable" a particular behavior of the system is for the application in hand. When defining this acceptability function, the system developer has to take into account the appropriate weightings of the different factors such as reliability, safety, performance etc. and give them the corresponding influence on the acceptability value. By moving the problem of weighting the different ingredients to a user-defined acceptability function a single, unified approach can be used to calculate and compare overall dependability measures.

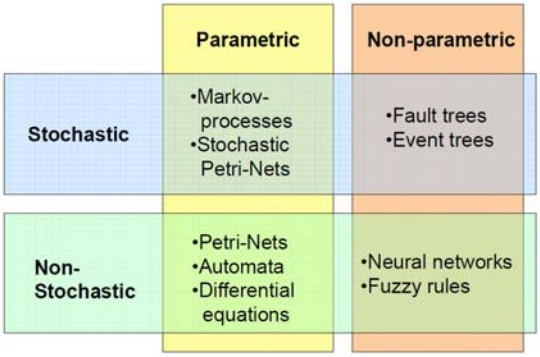
In contrast to the classical reliability engineering approach where the source of faults is not taken into account, in our approach we consider the behavior of dynamic systems. Therefore systems are described using models with uncertainty combining deterministic and stochastic processes. For reliability investigations, our assessment strategy is based on the hypothesis testing approaches commonly used in many disciplines, e.g. metrology and psychology, to determine the level to which a particular hypothesis is valid in a particular scenario. The problem of estimating a system's capability for a new application is then cast as the problem of establishing the likelihood that a given level of service (the hypothesis) will be delivered by a specific system in a given context based on the previous tests performed on that system. Another major advantage of the approach is that it lends itself to use of tests sheets [7] to define and apply the tests used to ascertain the dependability of a system and to document how the acceptability functions are used to calculate the final dependability measure [8].

In this paper we provide an overview of our Integrated Dependability Metric (IDM) approach and explain how system specifications can be enhanced with acceptability functions to combine dependability ingredients in an application specific way. This is demonstrated through a small case study for a dynamic control system.

## Theory

There are various different ways to describe dynamic systems, which can be classified as parametric/non-parametric and stochastic/non-stochastic models (see Table 1). In system theory, parametric input-output and state space models are very common, and are often enhanced by additional noise terms to model the stochastic part of the system.

**Table 1:** Model classification




This kind of model covers a broad range of applications. In contrast to pure stochastic models the output values are generally correlated, i.e. the random noise due a stochastic process is modified by the system transfer function. Thus, not only the distribution in the amplitude but also the distribution over the time domain is an important property of a stochastic variable. Furthermore deterministic output errors can be modelled as a result of parameter errors in the system transfer function. For physical systems, samples are directly related to the time at which the samples are taken, e.g. using a specified sampling rate. For other systems like software systems this sampling time is not obvious. Therefore it may be abstracted using an ordered series of samples instead.

Safety-critical physical states can be represented as internal states in the state space representation. However, as long they can be measured, critical states may be visible as output values of input-output systems. In order to fit the black-box view of many other disciplines, including software engineering the input-output representation for dynamic systems is preferred in this paper.

### Dependability Measure

The dependability measure defined in this paper is based on the definition of dependability for autonomous systems [5] and enhanced by an additional stochastic view of the system model. Furthermore, the validation against specified system properties plays a major role. The formal definition of our dependability measure makes the following assumptions:

1. The specification and the realization of a dynamic system are given: i.e. the purpose of the system (usage, mission), the behavioural, structural,

functional and non-functional properties, environmental conditions and system boundaries,

2. The system is operated in an environment with uncertainty, i.e. it is not exactly known if errors in the output behaviour are due to disturbances or system faults.
3. The correctness of known system properties is verified by other means
4. Faults will happen!

The required dependability measure *D* describes the correspondence of the actual system behaviour to its specified behaviour within the system boundaries and according to acceptance criteria. The dependability measure is an objective value and therefore free from any human perception and interpretation. The dependability measure is a functional, which depends on the actual system output behaviour *y*, the specified reference behaviour $y_r$, system boundaries and acceptance criteria $\Sigma$, a mission *u* (finite set of input test trajectories corresponding to the usage of the system) and a number of acceptability functions corresponding to the measured dimension of dependability.

As the Integrated Dependability Measure (IDM) for safety-critical computer controlled systems we propose the (time) discrete function:

$$D_l = 1 - \overline{D}_l = 1 - \frac{\sum_{j=1}^{d}\left[ a_j \frac{1}{m} \sum_{k=1}^{l} \overline{A}_j\left(u_k, y_{r,k}, y_k, \Sigma_j\right)\right]}{\sum_{j=1}^{d} a_j} \qquad \textbf{(1)}$$

with normalized acceptability factors $A_j = 1 - \overline{A}_j$ corresponding to the dependability component *j* with dimension *d* and the *k*-th sample of a mission of length *m*. For practical purposes, the acceptability factors $\overline{A}_j$ are normalized functions with values in the interval [0, 1]. The values are added using weighting factors resulting in the overall dependability function. Thus, dependability is a unique normalized value in the range [0, 1] (0 means undependable and 1 dependable) taking account of all possible system impairments during a mission. Both the actual behaviour and the reference behaviour are considered to be the system response on a set of predefined input trajectories called reference missions or usage profiles of the system. The reference behaviour represents the desired (expected) system response during the application of a specific input trajectory. Depending on the concrete system description the test inputs may be fixed trajectories or generated from a test pattern generator (e.g. test sheets or Markov-processes for stochastic systems). Prior to the execution of the test mission the system is initialized. In order to validate specified system properties, a set of criteria $\Sigma$ related to system behaviour and properties is explicitly included in the acceptability functions. Since performance, safety, complexity, etc. are often concurrent design parameters, the weights must be chosen by the system designer according to the system requirements.

**Acceptability factors for dynamic system performance**

The definition of errors is domain specific. For instance, in standard software technology only the correctness of a result is important and the error is modelled as a binary decision about the acceptance of the result. However in the scope of safety-critical real-time systems also the gradual degraded state must be considered.

As a basis function for several accessibility measures the relative deviation for sample $k$

$$e_{rel,k} = \frac{y_k - y_{r,k}}{y_p}, \tag{2}$$

related to the specified (maximum) error $y_p$ can be used. However, $e_{rel,k}$ may have unlimited positive and negative values. Therefore we propose the squared exponential function of $e_{rel}$ for the definition of an acceptability term for the system performance:

$$A_{performance,k} = \exp\left( -\left\| \frac{y_k - y_{r,k}}{y_p} \right\|^2 \right), \tag{3}$$

which has a range of [0, 1] and which is approximately $(1-e^2_{rel})$ for small values of $e_{rel}$. In case of a non-stochastic dynamic system this term reflects the structural and parameter uncertainty of the modelled system. If the system output is a vector $\boldsymbol{y}$ of dimension $d$ the Euclidian norm is used to determine the absolute value. In the case of stochastic systems $e_{rel}$ can be further evaluated by error statistics getting the meaning of reliability. Depending on how the system is described and what system property shall be highlighted (e.g. reliability or safety) the appropriate element of $\boldsymbol{y}$ must be chosen. In case of reliability, the output value is related to the service the system delivers. In the case of safety, the output value is related to critical system states (see example below). Furthermore, the test mission set must be carefully selected to cover the range of system outputs for the system properties under consideration.

**Reliability of dynamic systems**

For systems of high reliability the failure rate during normal operation is low. Generally, reliability parameters are determined using a large number of identical parts or many samples on one special system. However, for practical reliability evaluation of one system the lifetime may be shorter than the time needed to take the required number of samples.

The dependability concept presented here constitutes a generalisation of reliability and safety engineering concepts for dynamic systems. Consequently, the dependability measure should also include the special case of the reliability of static systems. Compared to established reliability measures using a binary fault model, here the gradual derivation of the system output can be used to reduce the testing effort.

In order to demonstrate this concept the system is modelled as a deterministic input-output system with stochastic uncertainty. The output value depends deterministically on an input stimulus, described by a constant transfer function known in advance. The output values are superimposed by a pure stochastic process

with independent samples. Thus, the stochastic process corresponds to the deviation of the actual output from the specified output. We assume that non-stochastic and stochastic process can be separated by subtracting both values.

**Special case: Reliability of dynamic systems with noise**

In contrast to static systems the output value of dynamic stochastic system depends not only on the actual input value, but also on the actual system state, which results from former input values and the initial value of the system. Generally the output values are correlated in the time domain, because independent input values, e.g. white noise, are modified through the dynamic system transfer function. Besides the probability density function (PDF) over the value range, the distribution of the output values over the time, respectively, the frequency spectrum, must be considered. Accordingly, the times where samples are taken must go along a system trajectory.

In this case we propose to collect output data from uncorrelated submissions and treat them as independent sample. To measure the proposed reliability factor it must be assured that the system did not change in between. For software systems, for example, this can be done by re-initialising the system.

**Special case: Reliability of stationary systems**

In this section the system under consideration is specified as a probabilistic system with the output variable *y*, statistically independent normal distributed output values, a mean value $\mu=0$, and a standard deviation of $\sigma$. Furthermore, it is required, that the absolute value of *y* does not exceed the maximum value $y_{max}$ with a probability of $P_{sys}$ for each sample of *y*. Thus $P_{sys}$ is a reliability measure for mean failures per invocation. The conditional probability

$$P(y \le y_{\max} \mid \sigma \le \sigma_0) \le \Phi(y_{\max}, 0, \sigma_0) \quad \textbf{(4)}$$

can be calculated using the cumulative distribution function $\Phi(y_{\max}, 0, \sigma_0)$ according to the normal distribution of *y*, $\mu=0$, and the specified value $\sigma_0$ for the standard deviation.

Consequently, the overall probability for an error free system is

$$P_{sys} = P(y \le y_{\max} \wedge \sigma \le \sigma_0) = P(y \le y_{\max} \mid \sigma \le \sigma_0) \cdot P(\sigma \le \sigma_0), \quad \textbf{(5)}$$

i.e. the product of the conditional probability of having no errors $P(y \le y_{\max} \mid \sigma \le \sigma_0) = 1-\gamma$ and the probability of being in the specified range $P(\sigma \le \sigma_0) = 1-\alpha$.

Accordingly, we can find an upper limit

$$\overline{P}_{sys} = \overline{P}(y \le y_{\max} \wedge \sigma \le \sigma_0) = 1 - P(y \le y_{\max} \wedge \sigma \le \sigma_0) \le \alpha + \gamma \quad \textbf{(6)}$$

for the overall error-probability (unreliability). Corresponding to the required reliability and the known PDF the $\sigma_0$ value is specified during system design. In order to test the system against its specification, it is not necessary to measure the absolute value of the reliability. The $\alpha$-value indicates the level of confidence the specified reliability has been reached. If the output samples are collected over a number of test missions,

$$\overline{A}_{reliability} = \alpha \tag{7}$$

can be used as an acceptability function for software reliability.

The system validation is now restricted to the test of the output value's distribution parameters, in this example the standard deviation, which can be performed using well known hypothesis tests. We assume that the PDF of the system output is known from long term experience, which is the normal distribution function in our case. Otherwise, tests known from textbooks can be used to test the PDF [9].

The hypothesis $H_0$: $\sigma^2 \le \sigma_0^2$ shall be validated by falsification of the counter hypothesis $H_1$: $\sigma^2 > \sigma_0^2$ with a confidence level 1-$\alpha$, i.e. the probability of accepting $H_0$ although $H_1$ is true is less than $\alpha$.

Since the real value of $\sigma$ is unknown, the sample standard deviation $s$ over a set of $n$ sampled output values $y_i$, $i=1..n$ will be used for further calculation.

For a given confidence level $\alpha$ = f($\chi^2_{n-1;\alpha}$) the test condition for rejecting $H_1$ is given by

$$\frac{(n-1)s^2}{\sigma_0^2} > \chi^2_{n-1;\alpha} \tag{8}$$

with the critical value $\chi^2_{n-1;\alpha}$ of the $\chi^2$-distribution with momentum $n-1$. The value $\chi^2_{n-1;\alpha}$ can be calculated according to the approximation formula

$$\chi^2_{m;\alpha} = m\left[1 - \frac{2}{9m} + y_\alpha\sqrt{\frac{2}{9m}}\right]^3 \approx m\cdot\left[1 + y_\alpha\sqrt{\frac{2}{9m}}\right]^3 \tag{9}$$

of Wilson and Hilferty with the critical value of the normal distribution $y_\alpha$. Since $\alpha$ decreases more than exponentially with $y_\alpha$, the number of samples required only increases weakly with decreasing $\alpha$ with

$$n - 1 \approx \frac{\frac{2}{9}y_\alpha^2}{\left[(s/\sigma_0)^{2/3} - 1\right]^2}. \tag{10}$$

In contrast, standard software verification techniques using the zero-error hypothesis need a sample size of the order O(ln$\alpha$/$p$) [10], which may be problematic for seldom events with a small probability $p$. This result is not surprising, because the proposed approach uses the complete information over the distribution of the output samples and not just the binary decision about the acceptance of the sample. Furthermore the assumption of a zero-error system is often not realistic and the measured confidence level corresponds to the error probability of the test and not to a system property. In case of the continuous model the $\alpha$–value corresponds to the confidence in the specified system and its parameters allowing an estimation of future system behaviour.

If the system output has a mean value $\mu \neq 0$ the complete PDF will be shifted on the y-axis resulting in an increase of the error probability P($\mu$)/P($\mu$=0)=1+$\phi$. For small

positive values of $\mu$ the error probability increases approximate linearly with $\phi=\mu \cdot y_{max}/\sigma^2$. Generally, $y_{max}$ and $\sigma$ are given by the system specification and $\mu/\sigma$ corresponds to the relative deterministic deviation of the system response $e_{rel}$, i.e. in the probabilistic case $e_{rel}$ has the meaning of a reliability decrease factor due to the non-probabilistic part of the system.

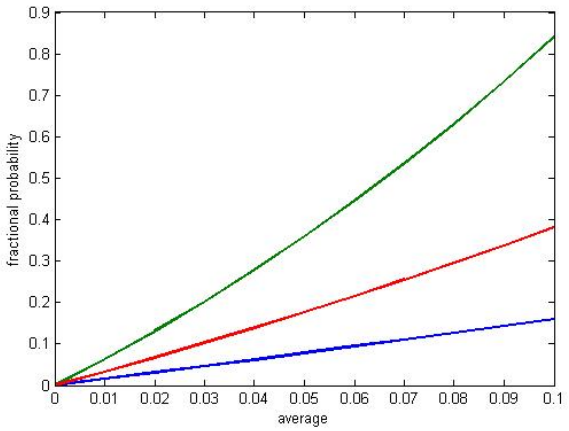



**Fig. 2.** Relative Increase of the error probability due to shift of the average value (blue line: $y_{max} = \sigma$, red line: $y_{max} = 3\sigma$, green line: $y_{max} = 6\sigma$)

**Safety of dynamic systems**

Generally, in the literature safety measures are probabilistic concepts based on binary accident events [3][4]. In order to find a safety measure describing the degraded state of a deterministic system the concept of the dynamic safety margin (DSM) is used [11]. In this concept the safety margin is the distance $\delta$ to the system boundary given by safety critical physical states, e.g. the pressure in a chemical reactor. In control engineering the measured DSM can be used in order to optimize or to adapt a controller during system operation.

Here, the DSM concept is generalized for safety-critical computer systems as a measure of how far a system is away from the critical state. Using an input-output description of the system, all safety-critical physical states must be accessible from outside the system or subsystem measured by the output value $y$. Similar to the performance acceptability a safety acceptability factor can be derived utilizing the DSM normalized to specified maximum deviation of the output value $y_S$.

We propose:

$$A_{safety,k} = \exp\left( -\left\| \frac{\delta_k}{y_s} \right\|^2 \right), \tag{11}$$

For dynamic systems with additional noise the DSM concept can be treated like the reliability concept, if we replace the maximum allowed deviation of the system output by the DSM. Thus the probability of reaching a critical state can be determined.

## Illustrative Example

In order to illustrate the concept of dependability measure a simple heating control system is described in this section. The heating system consists of a radiator, a switching controller and a temperature sensor. The controller gets the desired temperature $y_r$ from the input and the actual temperature $y$ from the sensor. If the difference of both temperatures leaves a specified range the controller switches the radiator ON ($u_k=1$) or OFF ($u_k=0$) corresponding to the control law:

```
initial value OFF
if (yr(k)-y(k))>0.1 && OFF than ON;
if (yr(k)-y(k))<-0.1 && ON than OFF;
```

The radiator temperature has an input-output behaviour corresponding to a first order system (low pass) with a time constant of 2300 s yielding the time-discrete transfer function

$$y_{k+1} = 0.9996 \cdot y_k + 0.3 \cdot u_k + 0.03 \cdot \nu_k \quad \textbf{(12)}$$

with the system input $u_k$ (power in watt) and the output $y_k$ (°C), normal distributed white noise $\nu_k$, and the sample period $\Delta t$. For simplification the index $k$ is used for all variables instead of $k\Delta t$, i.e. $y_k = y(k\Delta t)$. The specified deviation is $\sigma_0 = 0.15$°C. The maximum absolute temperature is $y_{max} = 40$°C. As a test trajectory a biased sinusoidal input is applied with $y_{r,k}=38.5°C+\sin(2\pi \cdot 0.001 \cdot k\Delta t)$, $m$=1000, $k = 0..m$, $\Delta t = 1$s, and initial output value $y_0 = 38$°C.

The acceptability functions are defined according to (**3**), (**7**), and (**11**) with $y_p=6\sigma_0$, $y_s=\sigma_0$, $\delta=y-y_{max}$. The dependability of dimension $d$=3 is defined according to (**1**) with the acceptability functions $A_1=A_{performance}$, $A_2=A_{safety}$, $A_3=A_{reliability}$, and weighting $a_1=0.3$, $a_2=0.4$, $a_3=0.3$.

The unacceptability of the reliability factor ($\alpha$-value) depends on the number of samples taken. In the test case $n$ = 100 samples are taken from independent runs, for one special instance of time. Corresponding to the low-pass behaviour of our system, the output deviation values are correlated in the short time range, leading to smaller $s$ values. In order to get uncorrelated samples, the system relaxation time must be awaited before the samples are taken. The time-dependent $\alpha$-values are shown in table 2, which increase with time.

**Table 2:** Sample of the time-dependent standard deviation of the system output and $\alpha$-value for $n$ = 100.

| $t$ (s) | $s$ (°C) | $\alpha$ |
|---|---|---|
| 10 | 0.1021 | 7.9134e-005 |
| 100 | 0.1029 | 9.6890e-005 |
| 1000 | 0.1047 | 1.5333e-004 |

The actual output and the reference output are shown in fig. 3a). The actual output has a noticeable noise in addition to the reference value. Correspondingly, performance acceptability fig. 3b), blue line, is also noisy and the corresponding cumulative function fig. 3b), green line, increases with number of samples accumulated.

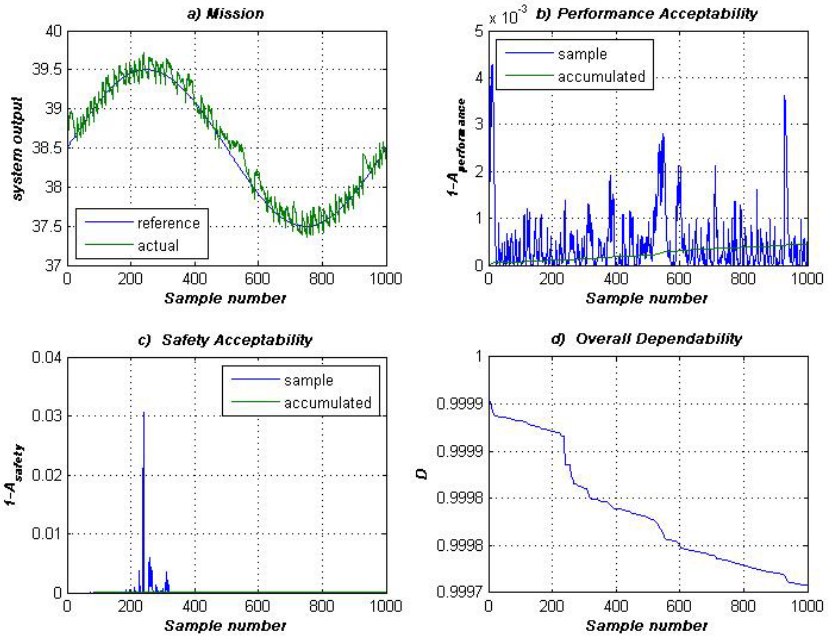



**Fig. 3. a)** System reference output (green line) and actual output (blue line); **b)** Performance (un)acceptability (blue line) and cumulative value (green line); **c)** Safety (un)acceptability (blue line) and cumulative value (green line); **d)** Overall system dependability.

Fig. 3c) shows the safety acceptability which has peak values in the sample range n=300..400. Within this range the system output has the minimum distance to the maximum system output (see fig. 3a). The overall dependability is plotted in figure 3d). It is obvious that the dependability decreases monotonically. The maximum slope is in the range where the weighted sum of all acceptability terms reaches its maximum as well. Therefore, the peaks in the safety function are visible as strong decreases of dependability.

## Conclusion

In this paper an Integrated Dependability Measure (IDM) for dynamic systems was proposed combining acceptability factors for different dependability relevant system properties. The approach is based on a behavioral system description which generalizes diverse system descriptions techniques from different disciplines, e.g. systems, hardware and software engineering, human factor engineering. The measure is suitable for stochastic as well as for non-stochastic system models and related properties. The measure is a functional of the specified behavior represented by the reference output trajectory, the actual behavior represented by the actual output trajectory and a specified mission represented by a test input trajectory. Furthermore criteria are defined by which the system can be validated. Thus, the dependability

measure does not describe the system behavior directly, but how much it deviates from the expected behavior.

The simulation example shows that using the dependability measure (1) the input-output behavior of a system can be validated against its specifications in relation to dependability requirements. As a special case, reliability metrics can also be included in the measure. Considering dynamic systems we have to project the system trajectory by reducing the time dimension to one single value in order to obtain stationary reliability values. Additionally, a method is proposed to validate the reliability by comparing the distribution of output values with a defined probability density function. This approach reduces the number of required samples significantly, which is necessary for a practical application. In this case, the absolute value of the failure probability is not required. In contrast, the deviation of the reliability from the specified value is measure indirectly. This also enables new testing methods to be used. In the future, the unified dependability measure will be applied and evaluated for additional systems covering typical examples of human factor engineering.

## References

[1] Laprie, J. C.: Dependability: Basic Concepts and Terminology. Ed. Springer, (1992).

[2] Avizienis, A. , Laprie, J.-C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. on Dependable and Secure Computing, 1 (1):11–33, (2004).

[3] Walter, M., Schneeweiss, W.: The Modeling World of Reliability/Safety Engineering. LiLoLe-Verlag GmbH, Hagen, Germany, (2005).

[4] Rakowsky, U. K.: System-Zuverlässigkeit, Hagen/Westfalen: LiLoLe-Verlag, Paperback, ISBN 3-934447-22-8, (2002).

[5] Rüdiger, J., Wagner A., Badreddin E.: Behavior Based Definition of Dependability for Autonomous Mobile Systems, in Proc. of the European Control Conference 2007, Kos, Greece, July 2-5, 2007, WeD11.4, (2007).

[6] Siewiorek, D. P., Swarz, R. S.: Reliable Computer Systems: design and evaluation. 3rd ed., A K Peter Ltd. Massachusetts, USA, (1998).

[7] Atkinson, C., Brenner, D., Falcone, G., Juhasz, M.: Specifying High-Assurance Services. IEEE Computer, vol. 41, no. 8, pp. 64-71, (2008).

[8] Atkinson, C., Barth, F., Falcone, G.: Measuring the Dependability of Dynamic Systems using Test Sheets, Workshop on the Design of Dependable Critical Systems, Hamburg, (2009).

[9] Weber, H.: Einführung in die Wahrscheinlichkeitsrechnung und Statistik für Ingenieure. Teubner, Stuttgart, ISBN 3-519-02983-9, (1992).

[10] Ehrenberger, W.: Software Verifikation: Verfahren für den Zuverlässigkeitsnachweis von Software, Carl Hanser, München Wien, ISBN 3-446-21624-3, (2002).

[11] Badreddin, E., Abdel-Geliel, M: Dynamic safety margin principle and application in control of safety critical systems. International Conference on Control Applications, volume 1, pages 689–694, Vol.1, 2-4, (2004).

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# Measuring the Dependability of Dynamic Systems using Test Sheets

Colin Atkinson, Florian Barth and Giovanni Falcone

Lehrstuhl für Softwaretechnik, Universität Mannheim,
68131 Mannheim, Germany
{atkinson, barth, falcone}@informatik.uni-mannheim.de

**Abstract.** Determining a system or component's dependability invariably involves some kind of statistical analysis of a large number of tests of its behavior under typical usage conditions, regardless of the particular collection of attributes chosen to measure dependability. The number of factors that can affect the final figure is therefore quite large, and includes such things as the ordering of system operation invocations, the test cases (i.e. the parameter values and expected outcomes), the acceptability of different operation invocation results and the cumulative effect of the results over different usage scenarios. Quoting a single dependability number is therefore of little value without a clear presentation of the accompanying factors that generated it. Today, however, there is no compact or unified approach for representing this information in a way that makes it possible to judge dynamic systems and components for their dependability for particular applications. To address this problem, in this paper we describe a new, compact approach for presenting the tests used to determine a dynamic system's dependability along with the statistical operations used to turn them into a single measure.

## 1 Introduction

Quantifying the dependability of software components and dynamic systems is a major challenge. In contrast with traditional "hard-wired" systems whose behavior remains fixed (or should remain fixed) as they execute, dynamic systems change their apparent behavior as time goes by – in other words, they remember the effects of previous operations and modify their behavior accordingly. According to this definition, most none trivial software systems and components are dynamic systems. Because of the memory effect, the dependability of dynamic systems cannot be calculated from a single metric derived by the repetitive application of a fixed evaluation criterion (e.g. MTTF from system failures or availability from system crashes etc.). Instead, the dependability of dynamic systems has to be determined from compound measures obtained by applying different evaluation criteria to the system's behavior using non-trivial scenarios resembling typical usage patterns. Only then does a dependability measure give a true estimate of the lilkihood that a dynamic system will deliver satisfactory service in a typical usage situation.

Intuitively, Dependability is a measure of the degree to which the users of a system can justifiably rely on the service it delivers – that is, its behavior. In general, there

are numerous properties or attributes of a system that influence its dependability, including [1][2]:

- *Availability*: the readiness for usage
- *Reliability*: the continuity of correct service
- *Safety*: the non-occurrence of catastrophic consequences on the environment
- *Confidentiality*: the non-occurrence of unauthorized disclosure of information
- *Integrity*: the non-occurrence of improper alterations of information

However, combining these separate factors into a single dependability measure is a highly application specific problem and there is currently no widely accepted theory that can be applied in a general way across different domains. To address this problem, the Ecomodis project has developed an approach to dependability specification and measurement that uses user-defined acceptability functions to provide an application-specific measure of a service's acceptability [3]. By observing a systems behavior over a series of carefully defined tests that mimic its real usage environment a picture of the system's overall behavior can be developed, and by using hypothesis-measurement statistics from such fields as psychology, a measure of a system's likely dependability for new applications can be obtained.

This approach relies on the clear and precise description of the tests used to exercise a system as well as the system's response to those test. However, traditional testing technology provides no concise way of describing such complex test scenarios or how the results of the tests are combined into higher-order measures. The most common way of doing this today is to write a software program in a general purpose programming language like Java or C++ that performs all the tests and applies the necessary statistical calculations to the results. However, just as with mainstream testing techniques based on standard software packages such as JUnit [4], this approach has a number of drawbacks. First, the ingredients and approach used are only understandable to software engineers who are familiar with the programming language used. Domain experts and managers who are unfamiliar with programming are unable to understand such descriptions. Second, even for people with the necessary expertise, the important information is obscured in a lot of superfluous programming "scaffolding" needed to create correct programs in the language concerned. This not only obscures the key test information and makes it more difficult to see, it also makes the task of writing correct descriptions more arduous and error prone.

To address this problem, the Ecomodis project has developed a new test specification technique to support the Ecomodis dependability model [3]. This approach, known as "Test Sheets" [5], was developed to combine the simplicity and readability of tabular test definition approaches such as FIT [6] with the flexibility of programmatic test definition approaches such as JUnit into a single unified approach based on the ubiquitous metaphor of spreadsheets. As well as allowing simple sequences of operation invocations (test cases) to be defined with the same expressive power as programming languages (but without the superfluous programming scaffolding) the approach also allows test case definitions to be nested to arbitrary

levels and parameterized in arbitrary ways. In contrast with programmatic approaches, test sheets can also describe the results of tests. To support the assessment of dependability, standard test sheets have been enhanced with (a) an additional set of columns which describe the satisfaction functions (in input test sheets) and the satisfaction values (in output test sheets) defined on operation input/output values, and (b) an additional row that allows statistical operations to be applied to these acceptability values and other values derived from the test. In this paper we provide an overview of this enhanced form of test sheet and explain the new features designed to support the measurement and specification of the dependability of dynamic systems. We first describe the basic principles behind test sheets and then show how they are used through a small case study.

## Expressing Usage Profiles with Test Sheets

The Test Sheet approach is a metaphor for test definition, application, and reporting which attempts to combine the power of programmatic approaches to testing with the readability and ease-of-use of tabular approaches. To achieve this goal a spreadsheet metaphor is used to identify the inputs to, and outputs from, operation invocations and express relationships between them. When complete, a language-independent test sheet can be transformed into source code in a specific target language for execution. Once executed, the test results can be visualized as a result test sheet. Furthermore, Test Sheets allow the definition of probabilistic or deterministic description of the test execution, thus allowing all kinds of behavioural protocols, algorithms [7] or any probabilistic operational profiles [8]to be defined.

To illustrate how test sheets support the process of measuring primitive dependability metrics and their combination into higher-order, compound metrics we use the example of a calculator. This component offers a number of mathematical operations that can be separated into two distinct groups that provide the basis for two different usage profiles:

- *basic operations*: add, subtract, multiply and divide
- *advanced operations*: log, sqrt, pow.

One usage profile characterizes applications that only use the basic operations of the calculator such as accounting applications. The other usage profile characterizes applications that also used the advanced operations such as scientific applications. Figure 1 and Figure 2 show the test definitions for the basic and the advanced usage profiles respectively.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 'Calculator | init | | | |
| 2 | E1 | add | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | C2+D2 |
| 3 | E1 | subtract | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | C3-D3 |
| 4 | E1 | multiply | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | C4*D4 |
| 5 | E1 | divide | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | C5/D5 |
| 6 | 100% -> 7 / 1 | | | | |
| 7 | 30% -> 7 / 2 | 30% -> 7 / 3 | 15% -> 7 / 4 | 15% -> 7 / 5 | 10% -> 8 |
| 8 | | | | | |

**Figure 1 Test Sheet for the basic usage scenario**

The Test Sheet in Figure 1 tests the calculator component using a basic usage scenario. The first line initializes the component, while lines 2 to 5 invoke the basic arithmetic operations with random values. More specifically, line 2 invokes the add operation (cell B2) of the calculator object returned from first operation (cell A2) with two random values uniformly distributed between 1 and 100 with a step width of 0.5 (cells C2 and D2). The result of the computation is compared against the sum of the two parameters to determine its correctness (cell E2).

The order in which these operations are invoked is defined by the behavioral information in lines 6 to 8 which represents a simple state machine. Execution starts with line 6. Cell A5 states that with a probability of 100% the control flow will be transferred to line 7 after performing the operation invocation in line 1, the initialization. Cell A7 to E7 define the relative probabilities of subsequent operations. If any of the cells A7 to D7 is selected, the execution state returns to line 7 after the corresponding operation invocation is performed. However, if cell E7 is selected (which has a probability of 10%) the control flow will be transferred to line 8, which is empty, thus terminating the test execution. The state machine represented by this test sheet is shown as a UML state diagram in Figure 3.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 'Calculator | init | | | |
| 2 | 'Helper | init | | | |
| 3 | E1 | add | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | C3+D3 |
| 4 | E1 | subtract | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | C4-D4 |
| 5 | E1 | multiply | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | C5*D5 |
| 6 | E1 | divide | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | C6/D6 |
| 7 | E2 | log | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | |
| 8 | E1 | log | C7 | D7 | E7 |
| 9 | E2 | sqrt | random uniform()[1...0,5...100] | | |
| 10 | E1 | sqrt | C9 | | E9 |
| 11 | E2 | pow | random uniform()[1...0,1...10] | random uniform()[-10...0,1...10] | |
| 12 | E1 | pow | C11 | D11 | E11 |
| 13 | 45% -> 14 / 1,2 | 45% -> 15 / 1,2 | 10% -> 16 | | |
| 14 | 30% -> 13 / 3 | 30% -> 13 / 4 | 20% -> 13 / 5 | 20% -> 13 / 6 | |
| 15 | 20% -> 13 / 7,8 | 30% -> 13 / 9,10 | 50% ->13 / 11,12 | | |
| 16 | | | | | |

**Figure 2 Test Sheet for the advanced usage scenario**

Figure 2 shows the test sheet representing the advanced usage scenario. In order to validate the return values of the advanced operations, a helper component is introduced that serves as a test oracle. The helper object is initialized in line 2 after the initialization of the calculator component. Lines 7 through 12 show how results returned by invocation of the advanced operations (lines 8, 10 and 12) are verified using results derived from operations of the helper component (lines 7, 9 and 11 respectively). In line 7, the log operation of the helper is invoked to obtain the value used to verify the result returned by the calculator's log operation. In line 8, that log

operation is invoked just like the basic operations in lines 3 to 6. However, in this case the input parameters are exactly the same as those used in the invocation of the helper component, indicated by the references to cells C7 and D7. The result of the calculator's log operation can thus be verified by comparing the value returned by the calculator to that returned by the helper component (cell E8). As before, the order of operation invocations is determined by the behavioural part of the test sheet in lines 13 through to 16. This is illustrated as a state diagram in Figure 4. In line 13 a decision is made whether to execute either a basic operation (line 14), an advanced operation (line 15) or to terminate the test execution (line 16) with the specified relative probabilities. In lines 14 and 15, the lines that represent the operations of the calculator are executed and the control flow is transferred to line 13 again, thus starting another loop through the algorithm.

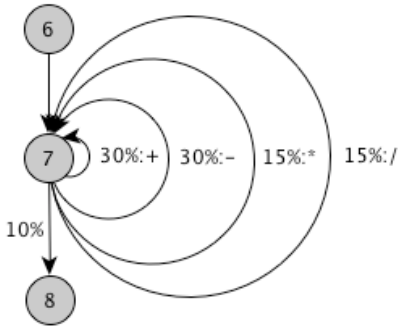



**Figure 3 State diagram basic scenario**

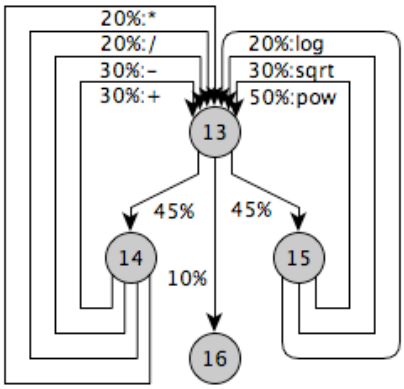



**Figure 4 State diagram advanced scenario**

## Test Sheet Extension for Dependability Measurement

The test sheets shows in the previous section are "standard" test sheets that can be used to define ordinary tests. Their strength is that by supporting the definition of behavioral information, components can be tested using realisitic, non-trivial scenarios. This provides the basis for obtaining meaningful dependability measures. However, it does not yet support the application of acceptability functions, nor the combination of acceptability values into higher-order measures. To support these, two further enhancements are introduced: acceptability cells and summary cells.

### Acceptability Cells

To support the application of acceptability functions, an additional column group called *acceptability cells* is added to the right side of the standard test sheet layout

(see Figure 5 Figure 7). These columns allow one or more satisfaction values to be calculated as defined by the equation or operation invocation in each cell. The contents of these cells can use the full expressive power of test sheets, like arithmetic expressions, cell references, etc. This allows the computation of complex metrics based on the runtime behavior of the component being tested.

Figure 5 shows the enhanced tests sheet corresponding to the simple scenario in Figure 1, illustrating the use of acceptability cells. The test sheets measures the acceptability of the results returned by the basic operations by computing the absolute delta between the returned and expected values. Line 2, in Figure 5 invokes the add method of the calculator component and the resulting value is stored in cell E2.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | 'Calculator | init | | | | |
| 2 | E1 | add | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | F2 > 0,99 | 1/(\|C2+D2-E2\|+1) |
| 3 | E1 | subtract | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | F3 > 0,99 | 1/(\|C3-D3-E3\|+1) |
| 4 | E1 | multiply | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | F4 > 0,9 | 1/(\|C4*D4-E4\|+1) |
| 5 | E1 | divide | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | F5 > 0,9 | 1/(\|C5/D5-E5\|+1) |
| 6 | 100% -> 7 / 1 | | | | | |
| 7 | 30% -> 7 / 2 | 30% -> 7 / 3 | 15% -> 7 / 4 | 15% -> 7 / 5 | 10% -> 9 | |
| 8 | | | | | | |
| 9 | errors | avg(F2:F5) | sum(F2:F5) | | | |

**Figure 5 Enhanced Test Sheet for Basic Usage Scenario**

The formula for the acceptability value first computes the delta between the returned and the expected result:

$$C2+D2 – E2$$

and then computes the absolute value of that delta:

$$|C2+D2 – E2|$$

This absolute delta is then put into a normalization function, in this case:

$$1/(x+1)$$

Figure 6 shows a plot of this function. The domain of the function is $[0,\infty]$ while the codomain is (0,1]. Hence the maximum value of the function is 1 for x = 0. The function is monotonically non-increasing so the value decreases for x-values greater then 0.

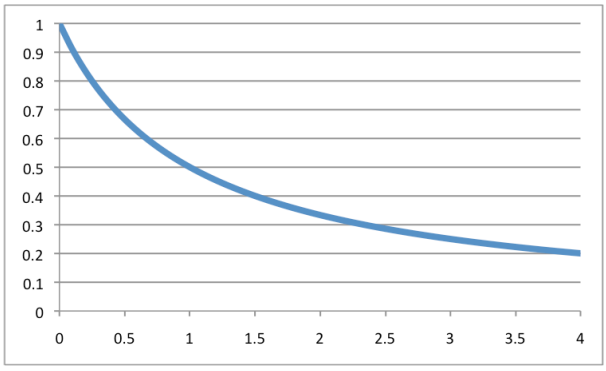



**Figure 6 Normalisation Function: 1 / (x+1)**

The complete formula is:

$$1/(|C2+D2-E2|+1)$$

Notice that this is only one possible acceptability function and the user is free to define computation formulas as needed. An additional option that becomes possible with acceptability cells is to use them to define the binary fail/pass criterion in the associated "Result Cells". In this case, the test will be marked as failed if the value of the satisfaction function is lower than 0.99 (cell E2).

Figure 7 shows the enhanced test sheet for the advanced usage scenario. In this case the acceptability of the advanced operation is calculated with a helper component. Instead of calculating the expected result inline as with the basic operations, the result of the helper component and the returned result of the calculator component are directly plugged into the function described before.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | 'Calculator | init | | | | |
| 2 | 'Helper | init | | | | |
| 3 | E1 | add | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | F3 > 0,99 | 1/(\|C3+D3-E3\|+1) |
| 4 | E1 | subtract | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | F4 > 0,99 | 1/(\|C4-D4-E4\|+1) |
| 5 | E1 | multiply | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | F5 > 0,9 | 1/(\|C5*D5-E5\|+1) |
| 6 | E1 | divide | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | F6 > 0,9 | 1/(\|C6/D6-E6\|+1) |
| 7 | E2 | log | random uniform()[1...0,5...100] | random uniform()[1...0,5...100] | | |
| 8 | E1 | log | C7 | D7 | F8 > 0,99 | 1/(\|E7-E8\|+1) |
| 9 | E2 | sqrt | random uniform()[1...0,5...100] | | | |
| 10 | E1 | sqrt | C9 | | F10 > 0,99 | 1/(\|E9-E10\|+1) |
| 11 | E2 | pow | random uniform()[1...0,1...10] | random uniform()[-10...0,1...10] | | |
| 12 | E1 | pow | C11 | D11 | F12 > 0,9 | 1/(\|E11-E12\|+1) |
| 13 | 45% -> 14 / 1,2 | 45% -> 15 / 1,2 | 10% -> 16 | | | |
| 14 | 30% -> 13 / 3 | 30% -> 13 / 4 | 20% -> 13 / 5 | 20% -> 13 / 6 | | |
| 15 | 20% -> 13 / 7,8 | 30% -> 13 / 9,10 | 50% ->13 / 11,12 | | | |
| 16 | | | | | | |
| 17 | errors | avg(F3:F12) | sum(F3:F12) | | | |

**Figure 7 Enhanced Test Sheet for Advanced Usage Scenario**

### Summary Cells

To allow high-order values to be derived from the information in the acceptability and result cells a new area containing the *summary cells* has been introduced beneath the definition of the behavior. These cells not only contain the formulas or invocations used to determine new higher-order values, they also represent return values of the test sheet for potential use in higher order test sheets. The keyword "errors" generates a list of cells that failed the check against the expected result. Similar to the formula notation supported by spreadsheets users may define arbitrary formulas for the calculation of further return values.

Summary cells enable the user to define computations that summarize the behaviour of the component during the test, thus allowing the definition of dependability metrics and the calculation of higher order measures in a consistent and readable way. Using higher-order test sheets that allow test sheets to be arranged in hierarchies, the return values of the test sheet invocations, i.e. the lower level summary measures, can be used for further computations. This allows different dependability measures to be further merged into a single compound measure. The test sheets in Figures 5 and 7 contain summary cells. In the case of the basic usage scenario (Figure 5) the first summary value is the list of failed cells, the second is the

average normalized deviation from the reference result, and the third is the normalized deviation. In the case of the advanced usage scenario (Figure 7) the same summary values are computed. Notice that in this case only cells that carry a value are relevant to the calculation, e.g. cell E7 will be left out when calculating the average or sum.

## Conclusion

In this paper we described how test sheet can be used to support the measurement and specification of system dependability, and presented two enhancements to standard test sheets introduced for this purpose. Because of tests sheets' ability to define behavioural information it is possible to test dynamic systems with realistic usage patterns, thus enabling the assessment of meaningful dependability measures. The first enhancement to standard test sheets is the introduction of a new column group to support the application of acceptability functions. These complement the result cells (that represent a binary decision on a test's success) using a continuous measure for the evaluation of the test results. The second enhancement is the introduction of a row group for the application of statistical operations to the test's return values. This facilitates the computation of compound measures and their presentation in a consistent and understandable way.

## References

[1] Melhart, B.; White, S., "Issues in defining, analyzing, refining, and specifying system dependability requirements," Engineering of Computer Based Systems, 2000. (ECBS 2000) IEEE Proceedings.
[2] Randell, B., "Dependability-a unifying concept," Computer Security, Dependability and Assurance: From Needs to Solutions, 1998. Proceedings , vol., no., pp.16-25, 1998
[3] C. Atkinson, A. Wagner and E. Badreddin, Towards a Practical, Unified Dependability Measure for Dynamic Systems, Workshop on the Design of Dependable Critical Systems, Hamburg, 2009.
[4] K.Beck. Test Driven Development: By Example, 2002.
[5] C. Atkinson, D. Brenner, G. Falcone, M. Juhasz. Specifying High-Assurance Services. IEEE Computer, vol. 41, no. 8, pp. 64-71, 2008.
[6] R. Mugridge and W. Cunningham, FIT for Developing Software. Framework for Integrated Tests, Robert C. Martin, 2005.
[7] H. Bär. Statische Verifikation von Softwareprotokollen. PhD thesis, University Fridericiana of Karlsruhe, 2004.
[8] J. D. Musa. Operational Profiles in Software-ReliabilityEngineering. IEEE Software. 10, 2 (Mar. 1993), 14-32, 1993.

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# Fault Propagation Analysis on the Transaction-Level Model of an Acquisition System with Bus Fallback Modes

Raul S. Fajardo Silva*, Jürgen Hesser, and Reinhard Männer

Department for Application Specific Computing, University of Heidelberg,
B6, 68159 Mannheim, Germany
{raul.fajardo@ziti.uni-heidelberg.de,
juergen.hesser@medma.uni-heidelberg.de}

http://li5.ziti.uni-heidelberg.de/

**Abstract.** The early fault analysis is mandatory for safety critical systems, which are required to operate safely even on the presence of faults. System design methodologies tackle the early design and verification of systems by allowing several abstraction for their models, but still offer only digital bit faults as fault models. Therefore we develop a signal fault model for the Transaction-Level Modeling. We extend the TLM generic payload by the signal characteristics: Voltage level, delay, slope time and glitches. In order to analyze and process these, a TLM bus model is created, with which signal faults can be detected and translated to data failures. Furthermore, inserting this bus in an acquisition system and implementing fallback modes for the bus operation, the propagation of the signal faults through the system can be assessed. Simulating this model using probability distributions for the different signal faults, 5516 faults have been generated. From these, 5143 have been recovered, 239 isolated and 134 turned into failures.

**Key words:** Signal faults, mixed signal verification, system design, fault modeling, system model

## 1 Introduction

Safety critical systems have to operate safely even on the presence of faults. It means that malfunctioning components have to be located and its faults isolated, so that it does not propagate to its user. The behavior of the system on the presence of faults can be analyzed using a model of the system. For that, faults and methods for localization, isolation and correction have to be modeled. In the

* Raul S. Fajardo Silva and Reinhard Männer are with the Department for Application Specific Computing. Jürgen Hesser is with the Institute of Experimental Radiotherapy, University of Heidelberg, Theodor-Kutzer-Ufer, 1-3, DE 68167-Mannheim, Germany.

case of an acquisition system the communication buses connected to the sensors are influenced externally by the environment and by each communicating node, being a critical point of the design.

The early design of complex hardware systems including software and hardware parts, interfacing with the real world and user is aided nowadays by system design methodologies. System design [1] abstracts the behavior of the system components by the specification of its function. In order to effectively design system communication, the Transaction-Level Model has been developed [2]. It allows the design of the communication to be independent from the components or architecture design. Furthermore the detail of the model can span from function calls to pin signaling.

In this paper we analyze the propagation of signal faults through a synchronous bus in a Transaction-Level model of an acquisition system. This system is composed by multiple sensors connected to a bus, a bus master and a CPU, which pools the data. First the bus, its modes and operating characteristics are modeled. The selection algorithm of fallback mode is placed on the communication controller, the bus master. For the fault injection, probability distributions are defined for the characteristics of the signal: Delay, slope level, voltage level and glitches, thus statistically generating faults. This faults are traced by the model so that their propagation results can be later evaluated.

The next section explains the bus model, its operating modes, the fault analysis and fault procesing modules. Section 3 presents the acquisition architecture simulated in this paper, the fallback selection algorithm and the fault generation, followed by the simulation results. In section 4 the conclusion of the work is presented.

## 2 Bus Model

In order to model a signal fault aware bus and its fallback modes, the TLM library is used. The actual standard considers performance issues related to the communication, but does not include operating characteristics to assure communication. The standard comprehends standard blocking and non-blocking transport interfaces and defines a standard payload[1] which includes performance characteristics, such as delay and latency [2]. We extend this standard payload to include the signal quality factors: Delay, slope level, voltage level and glitches, which are directly related to the bus operating capability. Furthermore, the model of the synchronous bus holds its operation mode: Operating frequency, clock phase and connected nodes.

Payload extensions contain both the value of signal characteristics of the transmission and a record of the violation of their limit (i.e. signal failure). When forwarding read calls, the extension is ignored, the signal characteristics are set by the callee. When the callee sends back the payload, the bus analyzes these, assigning the correspondent signal failure if the bus operation limits are

[1] An instance of the standard payload corresponds to a packet, when modeling a regular communication protocol.

exceeded. Furthermore the bus modifies the payload transmitted data, according to the occurred signal failures. At last, the complete payload is sent to the caller, fig. 1. Based on the failure record, the caller can then decide to change the operation mode in order to avoid further failure. On write calls, the bus first analyzes the signal characteristics and processes the data, then forwards these to the callee, ignoring the extension on return, fig. 1.

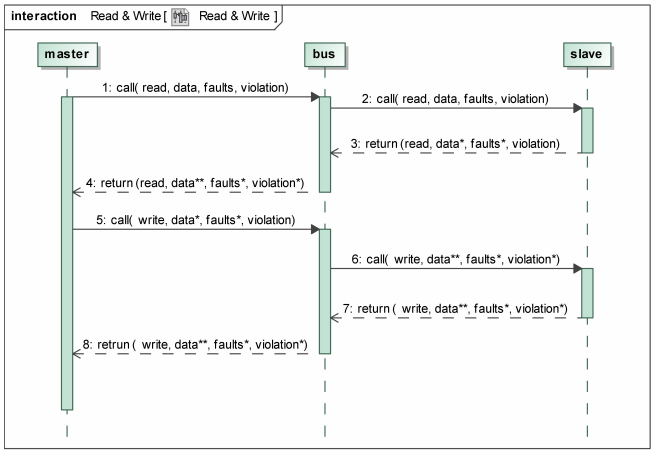



**Fig. 1.** Write and read calls to the bus (* represent that the variable has been set, ** modified)

### 2.1 Modeling Signal Faults

Prototype based communication monitoring techniques of [3] and [4] categorize bit faults, glitches and delays. [5] define possible signal faults of car sensors, as abnormal magnitudes (voltage levels), rolling (slope times), noise and dependency faults (context dependent). We categorize signal and bit faults related to digital hardware communication as a combination of both. A digital signal is ideally represented by two voltage levels, with instantaneous switching between both levels. For a real electronic component to drive its output from one logic level to the other, the resulting signal has a ***slope time***, which can be measured as a time degraded behavior. The same applies for ***delays***, which represent the response time of a component. Degraded ***voltage levels*** are variations of the output voltages for the logic levels approaching its boundaries, while ***glitches*** are voltage pulses of short duration resulting from interferences from outside. These four signal and bit faults of digital signals (fig. 2) are used as quality measurement of a transmission. These are then further divided for the characterization of a complete frame, composing the actual signal characteristics included in the payload extension: Both high and low bits voltage level; rise and fall time; delay; glitch time, level and count.

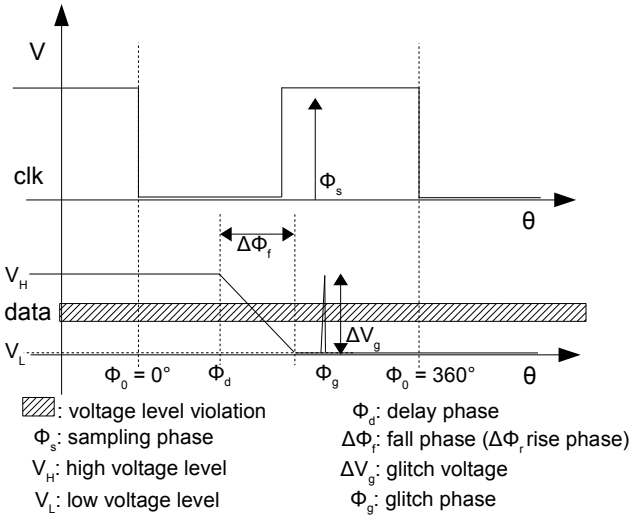



**Fig. 2.** Time normalized signal characteristics (time multiplied by operating frequency resulting in phase values)

| **Signal Conditions** | **Signal Failure Detection** | **Processing on Data** |
|---|---|---|
| $V_H < 2.0V$ | High bit Voltage Level | All 1s to 0s |
| $V_L > 0.8V$ | Low bit Voltage level | All 0s to 1s |
| $\phi_d > \phi_s$ | Delay | Rotate data to the right |
| $\phi_d + \phi_r > \phi_s$ | Rise time | Assuming $x[n]$ the series of the data bits $y[n] = \begin{cases} 0, x[n-1] = 0 \\ x[n], \text{otherwise} \end{cases}$ |
| $\phi_d + \phi_f > \phi_s$ | Fall time | Assuming $x[n]$ the series of the data bits $y[n] = \begin{cases} 1, x[n-1] = 1 \\ x[n], \text{otherwise} \end{cases}$ |
| Glitch count $> 0$ $\phi_s - 18° < \phi_g < \phi_s + 18°$ | Glitch time | Nothing |
| $V_H - \Delta V_g < 2.0$ | Glitch high level | If glitch time All 1s to 0s |
| $\Delta V_g + V_L > 0.8$ | Glitch low level | If glitch time All 0s to 1s |

**Table 1.** Signal conditions for signal failure detection (limit for bus operation) & Processing of the data according to detected signal failure (for the series, index 0 is bit 7 for a byte)

### 2.2 Fault Analysis and Processing

This module analyzes the signal characteristics of data being transmitted through the bus. The data sender sets the signal characteristics for the transmission. These are then compared to the conditions on table 1 to detect signal failures. The listed conditions are based on the limits imposed by the operation of the bus. For comparison, the timing signal characteristics are normalized to phase signal characteristics depending on the operating frequency of the bus. The bus operation conditions, sample time and clock phase are merged to the $\phi_s$ sampling phase. Logic levels and sample time are implementation dependent and thus constant, not influencing the relationship between operation mode and violation limits. Signal failures lead to data failure. In order to model that, the processes described in table 1 are carried out for each detected violation.

## 3 Acquisition Architecture

The architecture modules, acquisition CPU, bus master and sensors are modeled in SystemC using the Loosely-Timed coding style of the Transaction-Level Model, calling thus blocking transport only. The architecture connects the acquisition CPU to the bus master, which is connected to the sensors through the previously modeled TLM bus, fig. 3.

In the model of the acquisition CPU, only the acquisition pooling function is modeled. The bus master contains a thread safe buffer implementation, which is accessed by the CPU. To the other side it interfaces with the bus, executing two tasks. First, it request the data of every sensor. Then, if the bus detected a signal failure the bus master may change the operation mode of the bus and retry transmission. Furthermore, the operation mode of the bus can be periodically reset to raise bus performance, this also reconnects previously isolated nodes, which might have been faulty for a short period of time only.

Each sensor continuously reads data from a different input file, which can be accessed by calls to the blocking transport method. Upon each sensor access, the signal characteristics of the TLM extended payload are set. Despite of glitch count, these signal characteristics follow a Gauss distribution. The values for the mean and the standard deviation of the distributions can be set on sensor instantiation. The initialization value of the geometric distribution for the glitch count is equal the chance of no glitch occurrences in a bit. The statistical variable glitch count is then calculated by $framebits/x_k$.

### 3.1 Fallback Modes

In the bus master the fallback mode selection algorithm (fig. 4) can be activated. The bus master gets the information about signal failure occurrences from the bus instance. If the algorithm is activated and any failure occurs, a selected fallback mode is assigned to the bus by the bus master. Directly after mode change, a single transmission retry is carried out, for which neither fallback mode nor further retries are activated. After this transmission is completed, fallback modes can continue to be assigned.

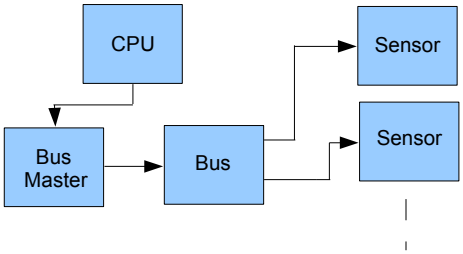



**Fig. 3.** Acquisition System Architecture

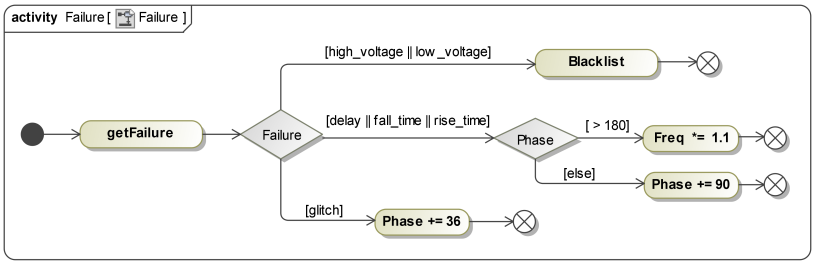



**Fig. 4.** Select algorithm for fallback mode

### 3.2 Results

During the simulation of the model all data is accepted by the acquisition CPU. Faulty data is marked on simulation and counted, if faults are detected, information about isolation or correction is logged, otherwise failure occurrence is asserted. With this data, fault propagation analysis can be made, producing statistics about the robustness of the model against the environment modelled by the probability distributions.

An environment is defined in the table 2. Bus works with a 100kHz frequency clock, sample phase of implementation 216°, and TTL logic levels (bit 0: 0.8 V/bit 1: 2.0 V). For a simulation on this configuration the values of total system faults (signal failures), fault isolation, fault recovery and failure occurrence are compared for 2 modes: Fallback reset on/off. Its results are presented in table 3. Mode fallback off does not isolate neither recover any fault, the same test for a fallback off bus produces the same amount of failures as arisen faults. The signal outputs of the data received by the CPU for a simulation with fallback turned off and on can be seen on fig. 5.

Applying signal fault detection and adapting the bus operation mode accordenly, 97% of the faults generated by the faulty behavior described in table 2 could be recovered, the remaining 3% have occurred on the retry transmission after fallback mode set. In this situation no further retry is activated.

| Signal Characteristic | Mean | Standard Deviation |
|---|---|---|
| High bit Voltage Level | 3 | 0.35 |
| Low bit Voltage level | 0 | 0.3 |
| Delay | $4\mu s$ | 1.2 $\mu s$ |
| Rise time | $2\mu s$ | $0.1\mu s$ |
| Fall time | $2\mu s$ | $0.1\mu s$ |
| Glitch time | $4\mu s$ | $0.5\mu s$ |
| Glitch level | 0.5 | 0.1 |

**Table 2.** Signal characteristic of sensor bus connection. Glitch count initialization value is 0.8, that is 80% chance of glitch free bit

| Number of | Fallback reset ON | Fallback reset ON |
|---|---|---|
| **Transmissions** | 40000 | 40000 |
| **Transmission Retries** | 2833 | 6 |
| **Blocked Transmissions** | 457 | 39198 |
| **Faults** | 5516 | 8 |
| **Isolated Faults** | 239 | 4 |
| **Recovered Faults** | 5143 | 8 |
| **Failures** | 134 | 0 |

**Table 3.** Test results for fallback reset ON and OFF tests

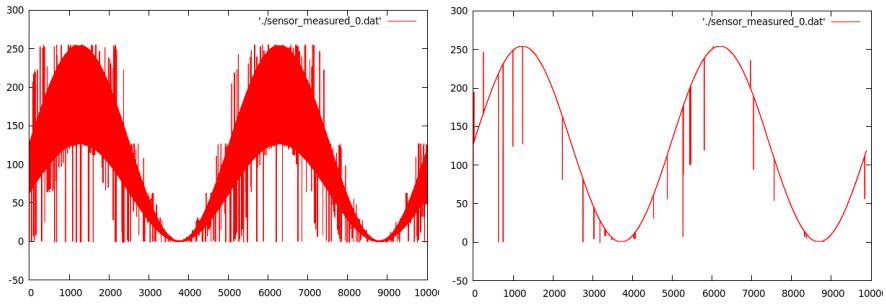



**Fig. 5.** Testing signal faults on bus: left fallback modes off, right on

## 4 Conclusion

The verification using classic hardware description languages evolves towards applying mixed signal verification to reduce uncertainty about the interoperability between analog and digital systems. Faults in the different abstraction levels of TLM have not been yet completely modelled. In this paper we have introduced a mixed signal verification strategy for TLM models, which profits from early verification of system design.

In order to process and analyze signal faults created in the system, we first developed a signal fault model, based on standard signal quality characteristics. Afterwards, an algorithm for detecting these faults based on operating properties of the same bus was created. Similarly, the same bus processes the transmitting data generating data failures according to the detected signal faults.

Then we inserted the developed bus in a TLM model of an acquisition system to reason about fault propagation through a bus with fallback modes. Here a bus master is implemented, which controls the bus, providing the bus with different operation modes. Faults have not been directly injected in the system. Instead, probability distributions have been assigned to the different signal characteristics of the sensors, building the environment of the system, which statistically generates faults.

The description of the signal characteristics of the sensors is realistic and can be easily adapted to different conditions. The online adaptation of the operation modes of the bus is able to isolate and correct almost every fault by sacrificing performance. In a future work we plan to compare this results with the fault tolerance of communication protocols with error correcting codes and error detecting codes with retries.

## References

1. Grant Martin, Brian Bailey, and Andrew Piziali. *ESL design and verification a prescription for electronic system-level methodology.* Morgan Kaufmann, 2007.
2. Open SystemC Initiative. http://www.systemc.org.
3. R. Pallierer, M. Horauer, Zauner M., A. Steininger, E. Armengaud, and F. Rothensteiner. A generic tool for systematic tests in embedded automotive communication systems. In *Proc. of the Embedded World Conference*, 2005.
4. E. Armengaud, F. Rothensteiner, A. Steininger, and M. Horauer. A method for bit level test and diagnosis of communication services. In *Proc. of the IEEE Workshop on Design & Diagnostics of Electronic Systems*, 2005.
5. J. A. Crossman, Hong Guo, Y. L. Murphey, and J. Cardillo. Automotive signal fault diagnostics - part i: signal fault analysis, signal segmentation, feature extraction and quasi-optimal feature selection. 52(4):1063–1075, July 2003.

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# The Impact of Individual Differences in Fine Motor Abilities on Wheelchair Control Behavior and Especially on Safety-Critical Collisions with Objects in the Surroundings

Meike Jipp, Christian Bartolein, Achim Wagner, Essameddin Badreddin

Automation Laboratory, University of Heidelberg, Germany
{meike.jipp, christian.bartolein, achim.wagner, badreddin}@ziti.uni-heidelberg.de

**Abstract.** In order to significantly reduce the number of safety-critical collisions of wheelchair users with objects spread in their environment, a study has been conducted which relates wheelchair user's fine motor abilities with the collisions while driving through a standardized course in a realistic office environment. The conducted inferential statistics demonstrate that especially the participants' aiming capacity can significantly predict the collisions occurring while driving through the course. A graphical and qualitative analysis of these effects demonstrates in addition that specific maneuvering tasks influence this relationship and that especially driving next to an object without colliding requires a high level of aiming capacity. The results demonstrate the need to develop a wheelchair system which adapts its assistive functionality to the aiming capacity and the difficulty of the maneuvering task in order to provide as much help as necessary without risking the degradation of the wheelchair user's skills.

**Keywords:** human-technology interaction, powered wheelchair control, fine motor abilities, adaptive automation systems

## 1 Motivation and State of the Art

The major goal of assistive technologies is to significantly ease the lives of those with sincere disabilities or serious impairments when executing activities of daily living. An example for such an assistive technology is an electrically powered wheelchair, which enables a mobility-impaired user to move freely and to a large degree independently. As a number of evaluations (see e.g., [1]; [2]; [3]; [4]; [5]) has demonstrated, this ambitious goal of easing the lives of those in need has not yet fully been achieved: While qualitative evaluations ([1]; [2]; [3]; [4]) demonstrated that long, tedious, and sometimes even unsuccessful training periods are required in order to use such an assistive device efficiently and effectively in everyday life; quantitative evaluations ([5]) showed that these (negative) effects can be traced back 1. to the number of input commands which are required in order to execute a given behavior, 2. to the space necessary for realizing special maneuvering tasks, and 3. to the time it

takes to actually reach the desired goal position. These statistics are even more sincere considering the number of accidents of wheelchair users occurring, e.g. when driving backwards without noticing a staircase behind them going down.

A number of wheelchair assistance systems have been developed in the past, which aim at improving today's technology for example by providing intention estimation behaviors and implementing methods developed in the field of robotics in order to automate as much as possible of the steering task (see e.g., [6]; [7]; [8]). This approach of easing the lives of those in need by taking over a great amount of the physical and cognitive work to actually control the assistive device is, however, criticized by physicians and nurses. The latter promote the concept that the assistance should only de-burden the persons with disabilities from those tasks, which cannot be achieved in their current condition, as otherwise the remaining skills and abilities deteriorate. Hence, as much support as necessary should be provided, not as much support as possible. In order to realize this vision, the development of an adaptive wheelchair system has been promoted (see e.g., [9]), which actually recognizes the current ability level of its user, derives an appropriate assistance level and actually uses this assistance level to support the user with disabilities as much as necessary such that on the one hand the remaining skills do not deteriorate and on the other hand the lives of those in need are eased and enhanced.

## 2 Problem Statement

In order to be able to actually realize such an adaptive wheelchair system, the current state of the art lacks a linkage between the ability profile of a wheelchair user and the occurrence of safety-critical situations.

## 3 Solution Approach

In order to fill this gap, a study was conducted, which is thoroughly described in the following sections.

### 3.1 Description of the Study

In order to relate the ability profile of a wheelchair user with the occurrence of safety-critical situations, 23 wheelchair users were asked to drive through a standardized course in a realistic office environment (for a floor plan, see Fig. 1). Within this office environment, five goal positions were identified and the participants were asked to drive from one of these five goal positions to the next. With repetitions, 14 goals had to be reached (for a detailed description of the course, see [10]). These course sections were defined such that reaching them required the participants to execute for wheelchair users difficult but also typical behaviors (such as e.g., turning on the spot, see [11]).

The wheelchair which was used for data collection is a powered wheelchair from Otto Bock Healthcare GmbH (type B600), which is thoroughly described in [12]. This wheelchair was equipped

- with a control PC, which was mounted underneath the seat of the wheelchair and used to record data (e.g. on the route taken during the course),
- a touchscreen for human-machine communication, which was, however, switched off in this study,
- a set of ultrasonic sensors, which can be used for realizing a collision avoidance behavior (see [12]) and which were also switched off,
- and a head-mounted eye- and headtracking system, which can be used to realize a gaze-based intention estimation behavior [12].

While driving, it was recorded for each of the 14 sections of the course and the complete course, whether and how often the participants hit objects such as tables spread in the environment. In addition, the participants' fine motor abilities were administered with the Motor Performance Test [13]. More specifically, data on the participants' tremor, their aiming ability, their wrist-finger speed, and their arm-hand velocity was collected on a number of standardized fine motor tasks. In addition, the participants filled in a biographical questionnaire to control additional variance of the dependent measures. This data covered e.g. the participants' gender, age, profession, experience in driving, etc.

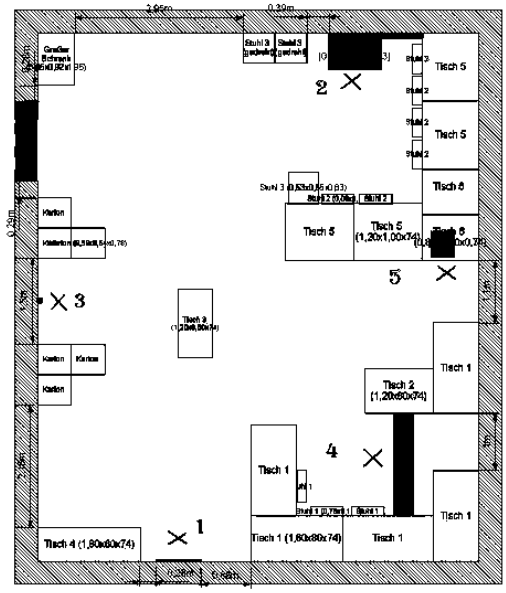



**Fig. 1.** Floor plan of the room in which the study took place – the crosses and the numbers inserted in the floor plan refer to the goal positions, which had to be reached by the participants.

Before the participants drove through the course, they were given unlimited time to practice with the wheelchair in the same environment in which the course was set up. This procedure was taken in order to ensure that no skill acquisition effects

influenced the data, as the participants were healthy individuals and have never been sitting in a wheelchair before in their lives. It was decided to work with healthy individuals due to practical considerations.

The sample consisted of 23 students of the Universities of Mannheim and Heidelberg (Germany). Most of them ($n$ = 20) were Bachelor students of psychology; the minority were Master's students ($n$ = 3) of computer engineering. The sample's average age was 23.1 years. 48% of the sample was male; 52% were female.

### 3.2 Data Analyses

After analyzing the descriptive statistics, inferential statistics were applied in order to relate the participants' fine motor abilities with the number of collisions when driving through the 14 sections of the course.

In a first step, univariate analyses of variance were conducted with the total number of collisions during the complete course as a dependent variable, the fine motor abilities of the participants as independent variables and variables such as the participants' gender as control variables. The analyses testing the relationship between (1) the tremor, the precision, the arm-hand velocity and the hand-finger speed and (2) the number of collisions were not significant. Significant results (see Tab. 1) were, however, found for the relationship between the results of the aiming capacity test and the number of collisions during the complete course:

- As Tab. 1 demonstrates, the time required to complete the aiming capacity task was a significant predictor ($F(1, 2) = 4.56$, $p < 0.05$, $f^2 = 0.19$) of the number of collisions caused while driving. As the reported statistics demonstrate, the effect is a large one according to the classification of Cohen [14]. As the positive correlation of $r = 0.26$ ($p < 0.05$) between the two variables demonstrates, the relationship is such that the greater the time required to complete the aiming capacity task, the more collisions occur.
- The other independent variables (i.e. the number of mistakes, the number of hits, and the duration of mistakes when completing the aiming capacity task) do not have a significant impact on the dependent variable ($p > .05$).

**Table 1.** Results of the univariate analyses of variance

| Independent Variable | Value of the test statistic $F$ | Probability $p$ | Effect size $f^2$ |
|---|---|---|---|
| aiming – number of mistakes | $F(1, 20) = 0.04$ | 0.71 | 0.01 |
| aiming – number of hits | $F(1, 18) = 2.41$ | 0.14 | 0.12 |
| aiming – duration of mistakes | $F(1, 21) = 0.06$ | 0.80 | 0.00 |
| aiming – total duration | $F(1, 20) = 4.56$ | 0.04* | 0.19 |

* $p < .05$

In a second step, general linear model analyses with repeated measurements were calculated using the number of collisions in each section as dependent variables, the fine motor abilities as independent, and variables describing additional information about the participants as control variables. In parallel to the results reported for the univariate analyses of variance, significant relationships were found mainly for the variables measured during the aiming capacity task. These significant effects are two-

way interaction effects between the repeated measurement factor (i.e., the number of collisions per course section) and the aiming capacity measure (i.e., number of mistakes, number of hits, duration of mistakes, total duration). More specifically, the following significant effects ($p < .05$) have been found (see also Tab. 2):

- The interaction between the repeated measurement effect and the number of hits explains a significant proportion of the dependent variable's variance with $F(13, 260) = 3.20$ ($p < .01$). Following Cohen's [14] convention, this effect size is large with $f^2 = 0.14$.
- The interaction effect between the repeated measurement effect and the duration of mistakes is significant with $F(13, 247) = 2.08$ ($p < .05$). In contrast to the previous effect, this effect can be considered medium-sized [14].
- Last, the interaction effect between repeated measurement effect and the total duration of the task is significant with $F(13, 247) = 2.63$ ($p < .01$). This effect is also a large effect ($f^2 = 0.12$).

**Table 2.** Results of the general linear model analyses

| Independent Variable | Value of the test statistic $F$ | Probability $p$ | Effect size $f^2$ |
|---|---|---|---|
| Aiming – number of mistakes | $F(13, 247) = 1.67$ | 0.07 | 0.08 |
| Aiming – number of hits | $F(13, 260) = 3.20$ | 0.00** | 0.14 |
| Aiming – duration of mistakes | $F(13, 247) = 2.08$ | 0.02* | 0.10 |
| Aiming – total duration | $F(13, 247) = 2.63$ | 0.00** | 0.12 |

* $p < .05$; ** $p < .01$.

In order to further analyze these effects, line plots were generated which are displayed in Figure 2.

These line plots first of all illustrate that the significant effects are mainly due to four sections of the course, which are Sections 2, 4, 7, and 14. These sections cover driving from Goal Position 4 to Goal Position 2; from Goal Position 5 to Goal Position 2; from Goal Position 3 to Goal Position 1 and from Goal Position 4 to Goal Position 1 (see Fig. 1). There is one criteria, which all of these course sections have in common, i.e., the goal position can only be reached if the participants drive next to an object: For Goal Position 2, the participants were asked to drive next to a cupboard such that they could withdraw a paper from it; for Goal Position 1 the participants were asked to drive next to a table. Hence, at least from this qualitative analysis of these course sections, it can be assumed that driving next to an object requires aiming capacity.

Second, the relationship between the performance in the aiming capacity tasks and the collisions was analyzed on the basis of these line plots. As the line plots demonstrate, the persons with worse aiming capacity performance collided more often in a course section, if they collided, when compared to those with better aiming capacity performance. In addition, the participants with greater aiming capacity performance measures collided less often within one course section; however, their probability of colliding overall sections was increased.

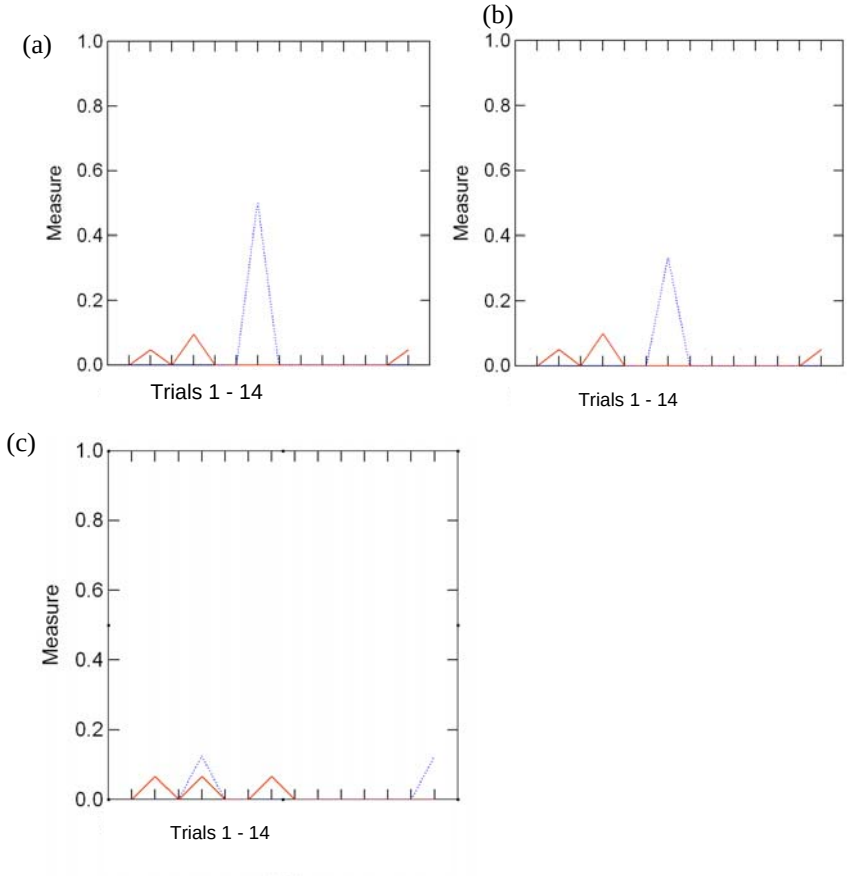



**Fig. 1.** (a) Line plot showing the number of collisions overall 14 course sections for those participants with an optimal number of hits (drawn-through line) and a worse number of hits (dotted line). (b) Line plot showing the number of collisions overall 14 course sections for the participants with greater durations of the mistakes (dotted line) and lower durations of the mistakes (drawn-through line). (c) Line plot showing the number of collisions during the 14 sections for those participants with a greater total duration of the aiming capacity task (drawn-through line) and smaller total durations (dotted line).

## 3 Discussion, Conclusions, and Future Work

It was the goal of this paper to demonstrate the relationship between the occurrence of safety-critical situations (i.e., collisions) and the fine motor abilities of wheelchair users. For this purpose, a study has been conducted, which is described in this paper, during which participants drove through a standardized course. Their collisions with objects in the environment were measured, as was their fine motor abilities.

The results of univariate analyses of variance and general linear model analyses demonstrate 1. a relationship especially between the aiming capacity performance measures of the participants and the number of collisions happening while driving through the complete course and 2. an interaction of this effect with the different sections of the course implying that there are maneuvering tasks, which require a higher level of aiming capacity than other maneuvering tasks. On the basis of graphical, qualitative analyses of line plots for participants with greater/lower aiming capacity performance measures and their collisions per course section, it was demonstrated that, on the one hand, participants with lower performance measures had an increased collision probability for some course sections requiring them especially to drive next to an object in their environment but a decreased collision probability for the complete course. On the other hand, the participants with greater aiming capacities collided less often during these risky sections, but had an increased risk of colliding during the complete course.

These results show that it is actually necessary to adapt the assistive functionality of a powered wheelchair system to the fine motor abilities (and especially the aiming capacity) of their users to successfully decrease the number of collisions with objects spread in the environment and to adapt the assistive functionality to the degree of difficulty of special maneuvering tasks in everyday behavior. As a next step, a cognitive model will be developed and implemented, which allows a wheelchair system to assess the aiming capacity level of its user and to adapt its assistive functionality accordingly (for a description of the methodology therefore, see for example [15].

## References

1. Bailey, D. M., DeFelice, T.: Evaluating movement for switch use in an adult with severe physical and cognitive impairments, American Journal of Occupational Therapy, 45(1), 76-79, 1991.
2. Bateni, H. Maki, B. E.: Assistive devices for balance and mobility: Benefits, demands, and adverse consequences. Arch. Phys. Med. Rehab. 86(5), 134-145, 2005.
3. Chase, J., Bailey, D. M.: Evaluating the potential for powered mobility, American Journal of Occupational Therapy, 44(12), 76-79, 1990.
4. Fehr, L., Langbein, W. E., Skaar, S. B.: Adequacy of powered wheelchair control interfaces for persons with severe disabilities: A clinical survey. Journal of Rehabilitation Research & Development, 37(3), 353-360, 2000.
5. Jipp, M., Bartolein, C., Badreddin, E.: Quantitative comparison of the joystick control mode and the two-switch control mode when steering a wheelchair. Accepted for Publication at the Annual Meeting of the Human Factors and Ergonomics Society, 2009.
6. Bartolein, C., Wagner, A., Jipp, M., Badreddin, E.: Multilevel intention estimation for wheelchair control. Proceedings of the European Control Conference, 1, 5463-5470, 2007.
7. Bell, D., Borenstein, J., Levine, S., Koren, Y., Jaros, A.: The navchair: An assistive navigation system for wheelchairs, based on mobile robot obstacle avoidance, Proceedings of the 1994 IEEE International Conference on Robotics and Automation. 1994.

8. Demeester, E., Nuttin, M., Vanhooydonck, D., Van Brussel, H.: A model-based, probabilistic framework for plan recognition in shared wheelchair control: Experiments and evaluation, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, Nevada, 2003.
9. Jipp, M., Bartolein, C., Badreddin, E., Abkai, C., Hesser, J.: Psychomotor profiling with Bayesian Networks: Prediction of user abilities based on inputs of motorized wheelchair parameters. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2009.
10. Jipp, M., Bartolein, C., & Badreddin, E.: Predictive validity of wheelchair driving behavior for fine motor abilities: Definition of input variables for an adaptive wheelchair system. Accepted for Publication at the IEEE International Conference on Systems, Man, and Cybernetics, 2009.
11. Kilkens, O. J., Post, M. W., Dallmeijer, A. J., Seelen, H. A., & Van der Woude, L. H.: Wheelchair skills tests: A systematic review, Clinical Rehabilitation, vol. 17, pp. 418-430. 2003.
12. Bartolein, C., Wagner, A., Jipp, M., & Badreddin, E.: Easing wheelchair control by gaze-based estimation of intended motion. Proceedings of the IFAC World Congress, 17, 9162-9167, 2008.
13. Neuwirth, W., Benesch, M.: Motorische Leistungsserie, Schuhfried, Mölding. 2004.
14. J. Cohen, Statistical power analysis for the behavioral sciences, Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.
15. Jipp, M., Bartolein, C., Badreddin, E., Abkai, C., & Hesser, J.: Psychomotor profiling with Bayesian Networks: Prediction of user abilities based on inputs of motorized wheelchair parameters. Accepted for Publication of IEEE International Conference on Systems, Man, and Cybernetics, 2009.

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# Real-Time Physiological Simulation and Modeling toward Dependable Patient Monitoring Systems

Ciamak Abkai[1] , Jürgen Hesser[1]
[1] Experimental Radiation Oncology, Mannheim Medical Center, University of Heidelberg, Mannheim, Germany
{Ciamak.Abkai, Juergen.Hesser}@medma.uni-heidelberg.de

**Abstract.** We present a novel approach to describe dependability measures for intelligent patient monitoring devices. The strategy is based on using a combination of methods from system theory and real-time physiological simulations. For the first time not only the technical device but also the patient is taken into consideration. Including the patient requires prediction of physiology which is achieved by a real-time physiological simulation in a continuous time domain, whereby one of the main ingredients is a temporal reasoning element. The quality of the reasoning is expressed by a dependability analysis strategy. Thereby, anomalies are expressed as differences between simulation and real world data. Deviations are detected for current and they are forecasted for future points in time and can express critical situations. By this method, patient specific differences in terms of physiological reactions are described, allowing early detection of critical states.

**Keywords:** Physiological Simulation, Real-Time, Risk Assessment, Patient Specific Modeling, Dependability

## 1. Introduction

Physiological modeling and simulation are very useful for various purposes in the medical domain (e.g. medical education, medical training simulators, interventional planning and understanding of physiological phenomena therein; as well as for prognostic modeling). Due to the multidimensionality of the problem, normally the overall modeling is a complex task (>4000 variables for quantitative circulatory physiology (QCP) [1]). In addition there are substantial uncertainties in the modeling data. Due to computational complexity, many approaches only apply population models and thus restrict to statistical information. Applying individualized physiologically based models including metabolism and transportation for different organs and tissues, however, allows for individualized simulations. Compared with population model based simulations, these individualized approaches are thus expected exhibiting the same advantages as we see them when comparing physiological based pharmacokinetic (PBPK) [2] with population pharmacokinetic (PopPK) [3] approaches.

We provide a new hybrid approach by combining stochastic modeling with integrative system, which provides realistic, patient individual and real-time capable simulations of physiological reactions to induced events e.g. given by medication or interventions. We present a novel methodology how approaches from system theory and dependability analysis therein can be applied to use real-time physiological simulations for patient risk assessment based on standard monitoring of high frequency physiological vital parameter addressing intelligent monitoring systems in clinical workspace.

## 2. State of the Art

One can find various micro and macro models considering special physiological interactions in human body. The main strategy focuses on using integrative models formulated by systems of ordinary differential equations (ODE) [4]. By Physiome [5] and QCP [1] a substantial step towards a platform for overall physiological modeling was established. Additionally, by these platforms it was possible for combine different smaller models into overall physiological descriptions and a general modeling language is supported, which allows building model data bases. The disadvantages are the lack of supporting real-time simulation, overcome model complexity issues as well as uncertainty of model parameters.

Thus, stochastic approaches are considered in our ap-proach as well. Especially dynamic Bayesian networks (DBN) [6] (as generalization of Markovian decision processes) are selected for medical simulations [7]. As shown earlier, the combination of integrative and stochastic approaches are well suited for real-time and realistic physiological simulations [8], and thus are essential as a basis for our risk assessment approaches.

System dependability, considered as a mixture of availability, reliability, safety, confidentiality, integrity and maintainability [9], is, unfortunately, not defined uniquely in literature and often it is system and mission specific dealing with errors, faults and failures. For dynamic systems, dependability is formally specified by the description of system behavior, such that the system trajectory remains in a certain predefined region/boundary [10]. Due to the fact that human factor is an important part of a monitoring system [11], diverse approaches consider the human in the context of dependability analysis [12]. Yet, no approach considers the patient's dependability additionally to technical systems so far. Even in the concrete field of patient monitoring, recent work on risk analysis only considers the system without patient [13]. We consider this as a systematic weakness, which we want to address and overcome with our new methodology by applying methods from systems theory in combination with dynamic simulations to provide a better and sophisticated way for risk assessment. The feasibility of our dependability strategy is demonstrated in a simulator environment extending a vital parameter monitoring system from the intensive care unit (ICU).

# 3. Methods

## 3.1. General Framework

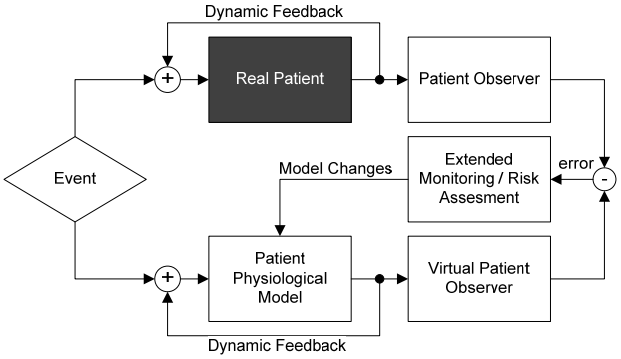



Fig. 1: System theoretical view: Event based real-time simulations have been used to simulate and predict the outcome of patient's vital signals, which can be measured/observed. The differences in signal outcome of the real and simulated patients have been used to describe dependability measures and provide an extended monitoring system.

Fig. 1 shows an extended patient monitoring represented in a system theoretic way. The upper part of the diagram shows *real patient* block, being a black box model and including some observable and non-observable internal states. This block describes the physiology (behavior) of the patient, in other words the patient's health states, which could be multiparametric. According to system dynamics – subsequent patient states are correlated to earlier ones – a *dynamic feedback* loop is necessary. As mentioned before, we are unable to observe and measure all patient internal parameters, which is depicted by a *patient observer* block. In the lower part of the diagram a corresponding network is found, which is a description of the virtual model, being a simulation model of the real patient. This system is, again, composed of *patient model* block, a *dynamic feedback*, and an *observer* block. The patient model may be any mixture of time-invariant dynamic systems even containing non-stationary probabilistic temporal models.

If the virtual model is mimicking/simulating the real world perfectly, there will be no difference in both observations. A difference, however, is interpreted as error given by the simulation, which – as depicted in the intermediate layer – allows extending the monitoring by providing more knowledge about patient states and even extend to patient dependability and risk analysis. Normally, if the virtual patient model is accurate and well suited, the error is a significant sign for a deviation between real patient states and virtual patient states. Such a deviation may be interpreted as a deviation from safety boundaries and hints towards possible safety critical situations.

### 3.2. Physiological Simulation Framework

As mentioned earlier, for the simulation engine a mixture of deterministic and probabilistic methods have been used, which provides better modeling capabilities especially by including stochastic causal influences, which can also have dynamic character. For this purpose, in addition to compartment models (Figure 2. left) DBNs (Figure 2. right) have been applied [16]. A DBN is a pair *(G,P)*, where *G* is a directed acyclic graph which nodes correspond to a set of random variables *x* of a stochastic time dependent process $X=\{X_t: t \in T\}$. $P=P(X)$ is the joint probability distribution (JPD) of variables of the random process X. Essentially, *G* describes the dependency by how far a variable is conditional or unconditional to other variables, i.e. a representation for causal influences between variables. The strength of influence is given by the conditional probability distribution CPD, which can be described for discrete and continuous space. For discrete space, the CPD can be specified by a finite conditional probability table (CPT), which is not restricting the CPD to predefined distributions e.g. a Gaussian. The main aspect of BN/DBN is the probabilistic inference, i.e. if the probability of a certain variable/node – called evidence variable/node – is known to affect the conditional probability of other variables/nodes. Various algorithms exist for performing exact inference, mainly based on applying Bayesian rule and d-separation on the JPD. On the contrary, approximate inference additionally supports large BN/DBNs and additionally operates on incomplete evidence in the network. In case of DBNs, the inference of nodes of future temporal slices corresponds to the prediction of future outcome and is therefore called temporal reasoning.

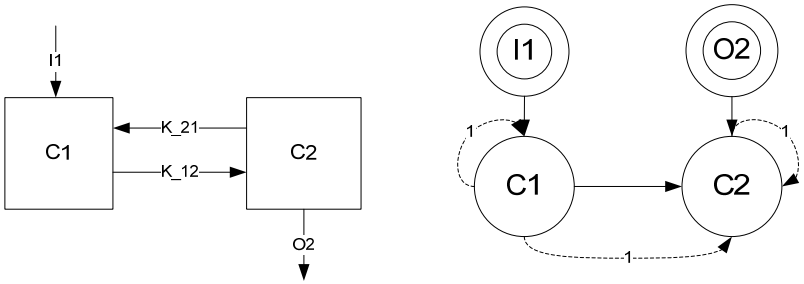



**Fig. 1.** Left: A 2-Compartment model. Right: The corresponding BN/DBN mixture containing static anchor nodes I1 and O2 from BN and two dynamic nodes C1, C2 from DBN.

### 3.3. Dependability and Risk Assessment Model

In clinical monitoring, a p*atient observer* (Fig. 1) analyzes and monitors patient's vital parameters, especially heart rate, blood pressure, oxygen saturation. Usually, these parameters are defined in a signal space *S*. By definition, monitoring devices adjust alarms, when a parameter exceeds a certain limit or boundary in the signal space. This procedure induces a subspace $\zeta \leq S$, where the signal is representing a non-critical and safe state. If $\zeta$ is time invariant with regard to the system dynamics it represents a constant interval, which is well-known from given alarm boundaries of

patient monitoring devices. We define the window dependability of a signal trajectory as shown in Eq. (1). $t_w$ is describing the time window of interest and $\varepsilon_\zeta^2(t)$ is the squared error given by the Euclidian distance of the signal value and a given boundary $\zeta$.

$$D_{t_w} = 1 - \frac{1}{t_w} \int_{t_0}^{t_0+t_w} \varepsilon_\zeta{}^2(\tau) d\tau \tag{1}$$

This formalism has two impacts; on the one hand the boundary $\zeta$ does not need to be a constant and on the other hand the integrative window shows how the boundary error is behaving over time. Additionally, dependability is defined with respect to a special mission [[9]]. In our case, stabilizing a patient's health state by an intervention or a medication is describing exactly such a mission and corresponding mission trajectories. For such a case, we define the mission dependability as given in Eq. (2). $t_m$ is describing the mission time which is given by the time for an intervention or a medication. $\varepsilon_\partial^2(t)$ is the quadratic error, which is given by the Euclidian distance of the real signal value and the simulated virtual signal value.

$$D_m = 1 - \underbrace{\frac{1}{t} \int_{t_0}^{t} \varepsilon_\partial{}^2(\tau) d\tau}_{past} - \underbrace{\frac{1}{t_w} \int_{t}^{t+t_w} \varepsilon_\xi{}^2(\tau) d\tau}_{future} \tag{2}$$

Hereby, one focus is on the dependability during a certain event based mission (from a starting time $t_0$ to an actual time $t$). The second focus lies on predicting dependability in future (from the actual time $t$ to the prediction horizon $t_w$). Thus, the formula consists of two parts; one error-formula for the past and one for the future. The error formula for the past can be interpreted on one hand as a measure for the quality of the simulation model. If the model is not simulating the real world accurately the error is large and the model is not well suited. By adding additional knowledge e.g. changing model parameters one adapts the model to the real world. This is either realized by user interaction or by applying multivariate optimization techniques. On the other hand if the model is designed well for healthy patients. The error term for the past is thus a good measure for the health state of a patient, taking time-variant information into account as well. Deviations to the health state is considered as reduced dependability like in system theory.

In our architecture, as shown in Fig. 1, we assume that there is a model which simulates and predicts the dynamic time-invariant changes of a monitored signal. Generally, such models are rare, because one needs to know the trajectory of the system states as well as the environmental influences. Therefore, probabilistic models are typically used to allow prediction of future system (in our case patient) states.

### 3.4. Quality of Service

According to our proposed architecture, it is possible to update the internal states of the dynamic system model by the knowledge of the real world observation. This process (which is called "smoothing" for probabilistic dynamic systems) will lead to

another prognosis for the next prognosis time window horizon $t_w$ [[13]]. Assuming that we can apply $N_w$ updates on the patient model within the time window $t_w$ will result in a measure for the quality of the predictions for future outcome, as shown in Eq. (3). The quadratic error $\varepsilon^{(i)}{}_{\partial}{}^{2}(t)$ is given by the Euclidian distance of the signal value and the predicted value $\partial^{(i)}(t)$ at time $t$ for $i=1..N_w$ model updates (smoothing) within in the prediction horizon. One has to consider that the entropy for probabilistic inference and thus the amount of uncertainty is increasing with the amount of reasoning steps $N_w$ and the prediction time $t_w$ [[14]]. Generally, in our terms this will lead automatically to worst quality of service for the predictive model.

$$Q = 1 - \frac{1}{t_w N_w} \sum_{i=1}^{N_w} \int_{t_w}^{t} \varepsilon^{(i)}{}_{\partial}{}^{2}(\tau) d\tau \tag{3}$$

## 4. Results

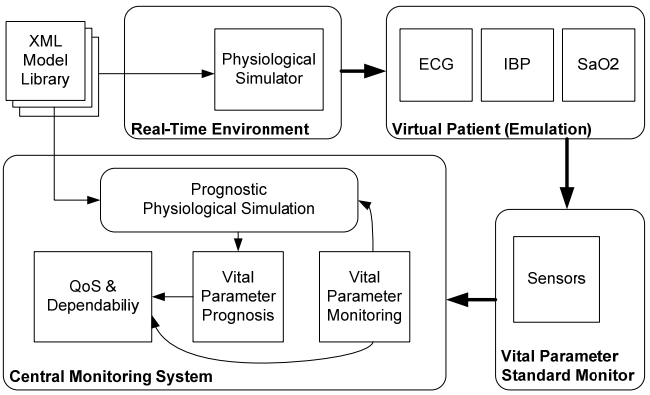



Fig. 2: Emulated vital parameter signals (ECG, IBP, SaO2) are detected by a monitoring system. An extended monitoring is supported due to the proposed methodology. Dependability and quality of service (QoS) are the major impacts of this method.

Our system developed for real-time-physiological simulations is using a hybrid approach applying ODEs and DBN for simulation of physiological interactions [5]. It is based on a hierarchical model description such that basic models for circulatory can be connected with e.g. models for drug interaction or interventional models as well. This system has been used to show the feasibility of the suggested approaches in a central monitoring environment.

We prepared a setup for a virtual ICU monitoring environment, as one can see in Fig. 3. A simulator dummy can simulate a real patient whose dynamics are represented by a set of models (e.g. circulatory system, medication, respiration defined in a XML model library) and patient specific parameters.

A similar simulation model is running virtually on the central monitoring system, while here the model parameters could be others. The virtual model updates internal states due to real measurements, emulated by the simulator dummy. The model

prognosis is analyzed regarding quality of service as well as dependability aspects for risk assessment.

In Fig. 4 we use a case study to show the feasibility of our methods on a medication with epinephrine, which is e.g used for the treatment of bardycardia. On the one hand a simulation (basic circulatory system in combination with simple 3-compartment PBPK) is running to forecast a prognosis for the effects on the heart rate (HR), on the other hand a similar simulation is running on the physiological simulator dummy to simulate the vital parameter in real-time. The measured data are processed by a monitoring system and emulate real data, although they are not from real patients. The error between forecasted and real data is used to compute the dependability value for the HR, given by the induced medication event. In fact, the error here is due to different parameter (clearance factor) given by the patient physiological model.

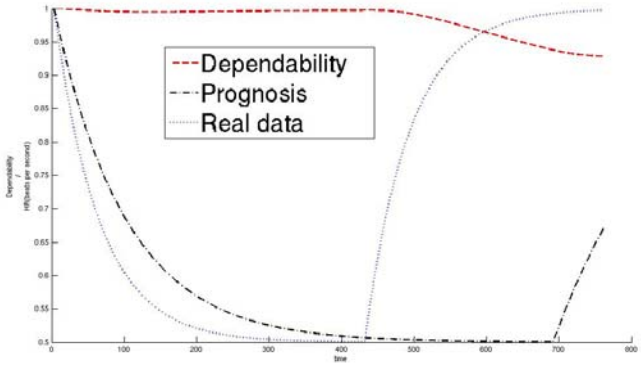



Fig. 3 Case Study: Effect of epinephrine on heart rate (HR) changes. One can see the forecasted HR due to the medication (Prognosis) and the real data extracted from the monitoring system. The error leads to a decreasing dependability value.

## 5. Conclusion and Future work

Applying dependability analysis on the human patient leads to interesting new methods for clinical monitoring. Physiological simulations are playing a key role in the proposed architecture, as far as they are addressed to take into account patient individual parameters as well as model updating and reasoning abilities. Once such models are available, the reasoning of events as medication or intervention for a specific patient based on the monitoring of vital parameter and other knowledge e.g. history, age and gender can be used for an individual risk assessment.

A general framework to access the dependability of patient states without forcing fault-tree modeling or similar approaches known from the reliability/dependability analysis have been provided by our methodology. The dependability measure for future risk and past model differences is a new view on patient's critical situations, which also considers dynamic attributes in addition to static ones, given by the well known alarm borders. Additionally the quality of service is a measure for the applicability of the virtual physiological model, which is currently in use.

We are preparing in vivo experiments on rats to test our methodology for vital parameter monitoring based on dedicational injection, showing how such a system

can be used to develop better and more specific models for drug interactions and provide a proof for the suggested concepts. By now, the applicability in terms of modeling and computational feasibility has been demonstrated as shown in Figure 4.

# References

[1] S.R. Abram, B.L. Hodnett, R.L. Summers, T.G. Coleman, and R.L. Hester, "Quantitative Circulatory Physiology: an integrative mathematical model of human physiology for medical education," *Advan. Physiol. Edu.*, vol. 31, Jun. 2007, pp. 202-210.

[2] L. Aarons, "Physiologically based pharmacokinetic modeling: a sound mechanistic basis is needed," *British Journal of Clinical Pharmacology*, vol. 60, Dec. 2005, pp. 581–583.

[3] Aarons, L. Population pharmacokinetics: theory and practice. Br J Clin Pharmacol. 1991, vol. 32, pp. 669–70.

[4] J. Keener and J. Sneyd, *Mathematical Physiology*, Springer, 2001.

[5] E.J. Crampin, M. Halstead, P. Hunter, P. Nielsen, D. Noble, N. Smith, and M. Tawhai, "Computational physiology and the physiome project," *Exp Physiol*, vol. 89, Jan. 2004, pp. 1-26.

[6] F.V. Jensen, *Bayesian Networks and Decision Graphs*, Springer, 2002.

[7] M.A. van Gerven, B.G. Taal, and P.J. Lucas, "Dynamic Bayesian networks as prognostic models for clinical patient management", *Journal of Biomedical Informatics*, vol. 41, Aug. 2008, pp. 515-529.

[8] C. Abkai, J. Hesser, "Virtual Intensive Care Unit (ICU): Real-Time Simulation Environment Applying Hybrid Approach Using Dynamic Bayesian Networks and ODEs", MMVR17, Studies in Health, Technology and Informatics IOS Press. Vol. 142. Jan. 2009. pp. 1-6.

[9] A. Avižienis, et.al. , "Dependability and Its Threats: A Taxonomy," *Building the Information Society*, 2004, pp. 91-120.

[10] J. Rüdiger, A. Wagner, E. Badreddin: "Behavior based description of dependability - defining a minium set of attributes for a behavioral description of dependability". ICINCO-RA (2) 2007: 341-346

[11] E. Coiera, "Intelligent monitoring and control of dynamic physiological systems," *AI in Medicine*, vol. 5, Feb. 1993, pp. 1-8.

[12] M. Jipp, C. Abkai, E. Badreddin, J. Hesser. Individual Ability-based System Configuration: Cognitive Profiling with Bayesian Networks. 2008 IEEE International Conference on SMC, pp. 3359 – 3364

[13] I. Maglogiannis, E. Zafiropoulos, A. Platis, and C. Lambrinoudakis, "Risk analysis of a patient monitoring system using Bayesian Network modeling," *J. of Biomed.Inform.*, vol. 39, 2006, pp. 637-647.

[14] M. Fisher, D. Gabbay, and L. Vila, *Handbook of Temporal Reasoning in Artificial Intelligence, Volume 1*, Elsevier Science, 2005.

[15] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[16] Gerven M.A., Taal B.G., Lucas P.J.: Dynamic Bayesian networks as prognostic models for clinical patient management. Journal of Biomedical Informatics, vol. 41, pp. 515--529 (2008)

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# An Integrated Monitor-Diagnosis-Reconfiguration Scheme for (Semi-) Autonomous Mobile Systems

Yi Luo, Achim Wagner, Leila Zouaghi, Essameddin Badreddin

Automation Laboratory, University of Heidelberg, Germany
{yi.luo, achim.wagner, leila.zouaghi, badreddin}@ziti.uni-heidelberg.de

**Abstract.** A nested monitoring, diagnosis and reconfiguration (MDR) scheme is proposed for a Recursive Nested Behavior based Control structure (RNBC) constituting a generic system architecture for (semi-) autonomous mobile systems. Each behavior layer within the RNBC structure is associated with a MDR schema, which is responsible to ensure the dependability of every single layer. An online dependability measurement and diagnosis procedure is integrated into monitor and diagnosis blocks under consideration of performance and safety acceptability factors. The reconfiguration blocks within the MDR-scheme switch from components with unacceptable behavior to redundant components, which may have degraded performance but more robust and safe behavior. The MDR blocks at each layer are nested through unified interfaces in order to utilize the distributed modeling of system behavior and to facilitate the system design and implementation process. In a small case study the MDR scheme is demonstrated for an assistant wheelchair on the body velocity control and axis velocity control levels. Simulation results show the feasibility and effectiveness of the approach.

**Keywords:** Dependability, autonomous mobile systems, monitoring, diagnosis, reconfiguration.

## 1 Introduction

In (semi-) autonomous mobile applications, the primary objective to use fault detection and diagnosis (FDD) and fault tolerant control (FTC) techniques is to increase system dependability. A unified FDD/FTC framework that adapt to behavior-based architecture is required to assist system development. Some research projects [1][2] have developed layered fault tolerant control architecture for behavior-based mobile systems. However, finding novel control structures and design methods which are better applicable to engineering applications are still important research questions in the field of fault tolerance [3][4].

This work proposes a nested monitoring, diagnosis and reconfiguration scheme, named as MDR scheme, which is designed for the Recursive Nested Behavior-based Control (RNBC) structure [5]. Fault modeling and dependability concepts are adapted from [6] and [7]. In contrast to binary fault modeling the dependability concept is based on the behavior description of the system and its components. Dependability

properties are related to the deviation of the actual system behavior from the desired behavior and to the distance of critical system states from safety boundaries. The desired behavior can be described in the form of a reference output signal (reference mission), which may be generated by a reference model in response of a system input trajectory. The deviation of the actual system output from the reference output is monitored by the corresponding monitoring component. If a state-space reference model is available, the monitoring component may be realized in form of a state observer, which estimates the internal system states besides the next predicted output value. The monitoring component outputs the deviation signals (residuals) and the distance of critical states from their limits. In case of black box modeling all critical states must be visible as external signals. The external signals will be fed to a diagnosis component, which assesses the acceptability of the retrieved value (s. example below). Depending on the result the system is reconfigured using a reconfiguration component. Here a hierarchical monitoring, diagnosis and reconfiguration (MDR) scheme is proposed.

## 2 Proposed MDR Concept

The MDR scheme is integrated in the Recursive Nested-Behavior-Based Control (RNBC) structure consisting of a number of layers, which are recursively connected to each other [5]. Each layer in the RNBC structure hosts a number of components and corresponding dynamic behaviors. The behaviors can be uniformly described as signals, which flow between the layers, regardless their type of implementations (e.g. hardware or software). A single MDR block ensemble is locally associated with a single behavior layer and responsible to keep the deviation from the specified behavior in an acceptable level.

Figure 1 shows two behavior layers of the RNBC structure, each of which containing a MDR scheme besides the functional components. Monitor, diagnosis and reconfiguration are the three components under consideration. The working principle of them will be explained in the following, using an exemplary modeling approach, i.e. all layers are described as linear time-invariant systems with time-continuous dynamics.

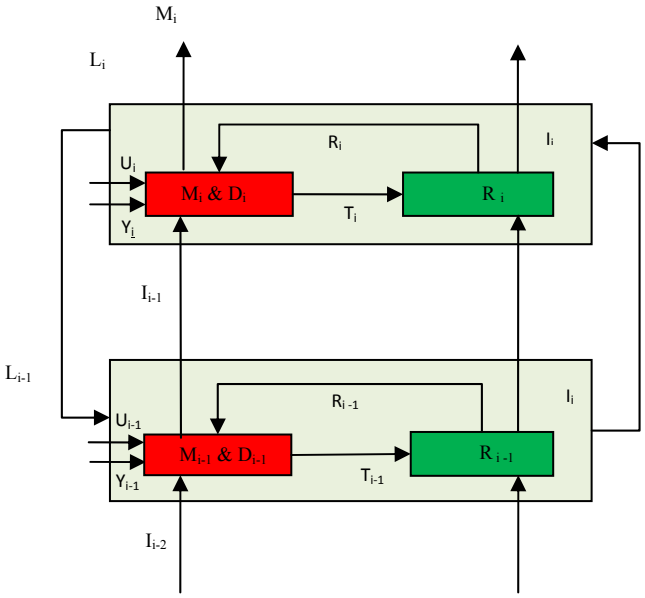



**Fig. 1.** Monitoring (M) – Diagnosis (D) – Reconfiguration (R) scheme integrated into the RNBC structure, exemplary shown for two behavior layers

### 2.1 Monitor Block and Diagnosis Block

The aim of the monitor block is to calculate the behavior deviation and the safety margin. Inputs for monitor $M_i$ are: measured ($u_i$, $y_i$) of the ith layer, lower monitor status information $I_{i-1}$, and reconfiguration information $R_i$ to indicate the status of the reconfiguration process and therefore to update the current reference model. A reference model, e.g. using a transfer function, which describes the nominal behavior of the considered layer, is required. The model is used to determine, for a given input, the reference output $y_{ref}$. The instantaneous deviation from the reference behavior is given by the residual (see also Fig. 2)

$$\boldsymbol{\varepsilon}_P(t) = \left|\mathbf{y}(t) - \mathbf{y}^{ref}(t)\right| \quad \textbf{(1)}$$

The residual is a basic ingredient for a normalized performance acceptability function

$$A_P(t) = 1 - \frac{\boldsymbol{\varepsilon}_P(t)}{E_p} \quad \textbf{(2)}$$

yielding a value range [0, 1] and indicating, how acceptable the system's (component's) behavior is in comparison to a maximum allowed output deviation $E_P$.

The definition of a safety acceptability function is also based on a behavioral description. Therefore, the concept a dynamic safety margin [6][8] has been adopted. In [8] safety boundaries for a state space model and a dynamic safety margin, which is the minimum distance from these boundaries, have been defined. In contrast to the original definition, here, the safety boundaries are related to the output signal, which

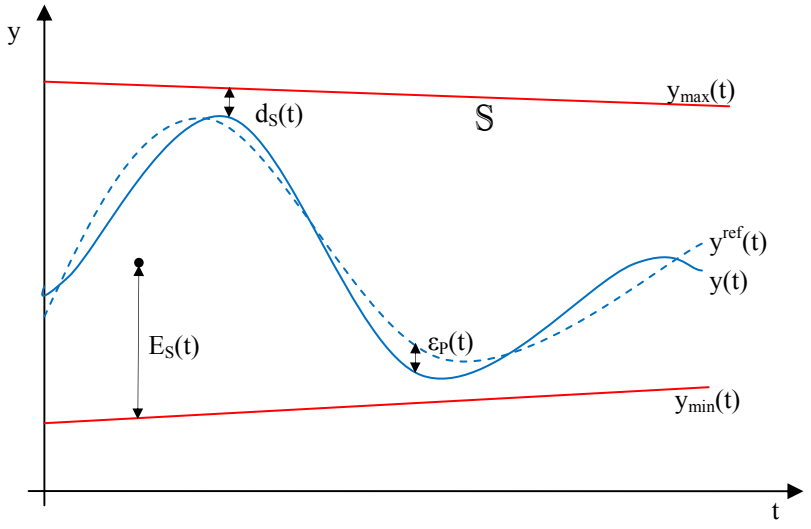



**Fig. 2.** Safety boundary and dynamic system trajectory.

is equivalent in the case of having all internal critical dynamic states available as system outputs.

For the given input u(t), there is a range $[y_{min}, y_{max}]$ for the output y(t), where the system is considered to be in a safe condition. Now let

$$d_s(t) = \begin{cases} \min(((y(t) - y_{\min}), (y_{\max} - y(t))), y(t) \in [y_{\min}, y_{\max}] \\ 0, y(t) \notin [y_{\min}, y_{\max}] \end{cases} \quad (3)$$

be the distance to the safety boundary (fig. 2) and

$$E_S(t) = \frac{1}{2}(y_{\max} - y_{\min}) \quad (4)$$

the centre point of unsafe region at time t. Now, we can define the safety acceptability function

$$A_S(t) = \frac{d_s(t)}{E_S(t)} \quad (5)$$

with $A_S(t) \in [0,1]$ reflecting the system (component) safety level with respect to the maximum possible distance to the safety boundary $y(t) = E_S(t)$.

The total acceptability is the weighted sum of all acceptability terms

$$A_{TOT}(t) = a_P A_P(t) + a_S A_S(t), \quad (6)$$

which is a function of time and which reflects the coincidence of the actual system behavior with the specified behavior. According to [6], the integration of the acceptability values over the system's mission trajectory leads to a unique overall

dependability measure. In this paper the instantaneous total acceptability function is used to decide, if the system yields an acceptance level $A^*$ or not. If $A < A^*$, a system reconfiguration is enabled.

### 2.3 Reconfiguration Block

There are basically two questions, which must be answered before a system reconfiguration can be performed: 1. What configuration should the system have after the reconfiguration, 2. How can the system be brought to the new configuration (especially how does the system behave during the transient phase).
The question, what new configuration shall be used can be answered as follows:

**Offline Design:** Each behavior layer contains a nominal components and redundant components. Both are designed and tested offline. E.g. the nominal component is designed to deliver better performance while the redundant component is simple, well understood, and more robust against faults. Thus, for each component the (average) acceptability value for a set of predefined typical mission trajectories can be measured during system test. During operation of the system the best component (with the highest acceptability level) is selected. The component (or even a complete layer) under consideration is replaced by switching if the acceptability level drops under the level of the next best component. It is required that all possible combinations of components behave stable. The offline design method proposed is in contrast to online design methods, where the complete system (structure and parameters) is rebuild according to the instantaneous system constraints.

The second question cannot be answered so easily, if the system can be switched forward and switched back between different (at least two) configuration, since the system may behave unstable even in the case, when the single configuration themselves are stable. Therefore, we assume here one single transient from an undependable configuration to a new dependable configuration.

**Online Switching:** By default, all nominal components are supposed to be “normal”. The switching is enabled only after the switching condition (enable signal from M&D blocks active) is fulfilled. When the reconfiguration is enabled, the reconfiguration block checks the configuration $R_i$ from lower layers and it checks then the stability of the redundant component in the loop with the lower layers. When the stability condition is fulfilled the switching process will start. If the redundant component is already in operation and detected to be failed, the whole system will be brought in a fail-safe condition.

## 3 Application of the MDR Concept to an Intelligent Wheelchair System

In this section, a small application scenario is proposed illustrating the concept of the MDR scheme. Therefore, the three lower levels of a human-assisting “intelligent” wheelchair control system are considered.

Figure 3 illustrates the MDR scheme for the velocity controller in a wheelchair system. Layer L1 consists of the axes-level velocity controller, the actuators and some data processing blocks. Layer L2 contains the body velocity controller. It gets the reference velocity $[x', \theta']^{ref\ 2}$ from layer L3, compares it with the measured velocity $[x', \theta']^{m\ 2}$ from gyroscops and encoders and generates a control signal, which is the reference velocity $[x', \theta']^{ref\ 1}$ for next layer. In this example, the primary PID (proportional-integral-derivative) controller is used as nominal controller. The secondary PI controller has lower performance but is simpler and more reliable. The secondary component is running in parallel with the nominal component (hot standby). Thus an initialization period and a long term transient phase can be avoided.

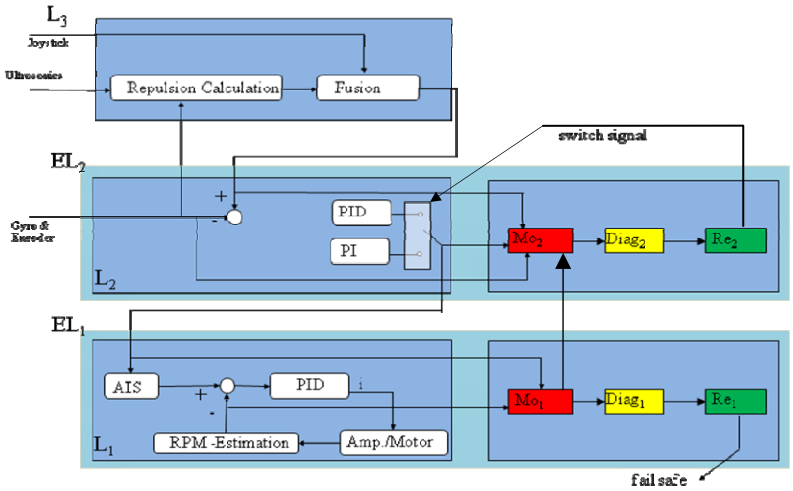



**Fig. 3.** Application example: Three lower layers of an intelligent wheelchair system

Parameters of PID/PI controllers and MDR are given in Table 1. These parameters comply with manufacturer and empirical data so that future implementation can be made based on them.

**Table 1.** System parameters for the body velocity control level.

| Components | Parameters | Value |
|---|---|---|
| PID Controller | $K_{p,trans}, K_{i,trans}, K_{d,trans}$ | 1.33, 1.11, 0.37 |
| | $K_{p,rot}, K_{i,rot}, K_{d,rot}$ | 1.6, 1.33, 0.53 |
| PI Controller | $K_{p,trans}, K_{i,trans}$ | 1.0, 0.5 |
| | $K_{p,rot}, K_{i,rot}$ | 1.2, 0.6 |
| MDR | $E_S$, $E_P$ | $[6, 6]^T$, $[5, 5]^T$ |
| | $a_S$, $a_P$ | 0.5, 0.5 |
| | $A^*$ | 0.75 |

Simulation results of the developed MDR mechanism using the model above are shown in Figure 4 a, b. A fault in layer L2 is emulated by injecting a 1 second output

delay in the nominal PID controller. Figure 4.a shows the L2 behavior in 3 cases. The Blue line corresponds to the faultless case, the red line denotes the faulty case without MDR scheme and the green line denotes faulty case with MDR scheme. Figure 4.b shows the time-dependent acceptability level during a mission of 100 seconds. As the desired acceptability level is 0.75, the behavior switching happens at t = 0. It can be observed that the MDR mechanism has recovered the behavior to an acceptable level by switching to the redundant component upon failure detection.

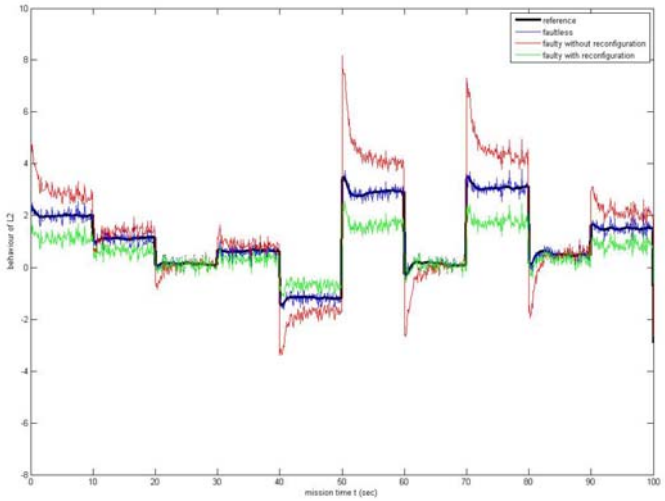



**Fig. 4 a.** L2 translative velocity behavior in faultless, faulty (no reconfiguration) and faulty (with reconfiguration) cases

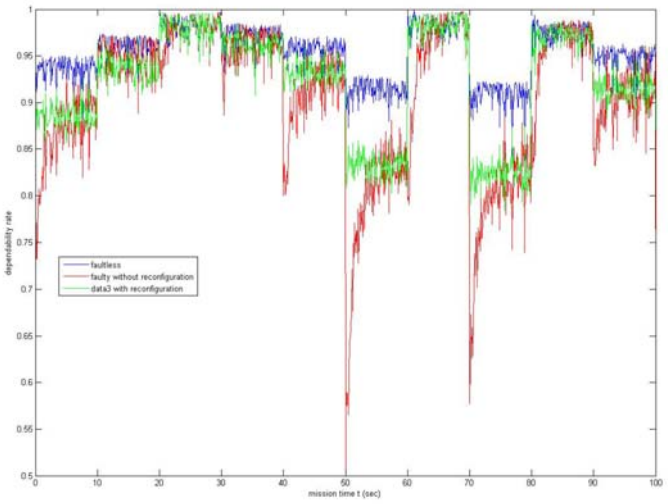



**Fig. 4 b.** L2 acceptability level in faultless, faulty (without MDR) and faulty (with MDR) cases

## 4 Conclusions and Future Research

In this paper, a monitor-diagnosis-reconfiguration scheme for autonomous and semi-autonomous systems is proposed. A Model-based monitoring and multiple-controllers online switching approach is realized and demonstrated within a realistic simulation example. Dynamic behavior acceptability improvement as reconfiguration goal is carried out. As a single behavior, the body velocity controller of a wheelchair system, was integrated together MDR within the proposed architecture. The simulation results show the feasibility of the proposed MDR scheme in terms of keeping the behavior of components and system layers within an acceptable performance and safety. In future research, the MDR scheme will be implemented into a real-time control system.

## References

1. Ferrell, C.: Failure Recognition and Fault Tolerance of an Autonomous Robot, Adaptive Behavior, vol. 2, no. 4, pp. 375-398 (1994).
2. Visinsky, M. L., Cavallaro, J.R., and Walker, I.D.: A Dynamic Fault Tolerance Framework for Remote Robots, IEEE Transactions on Robotics and Automation, vol. 11, no. 4, pp. 477-490 (1995).
3. Zhang, Y., Jiang, J.: Bibliographical review on reconfigurable fault-tolerant control systems, Annual Reviews in Control Volume 32, Issue 2, Pages 229-252 (2008).
4. Duan, Z.H., Cai, Z., Yu, Z.: Fault Diagnosis and Fault Tolerant Control for Wheeled Mobile Robots under Unknown Environments: A Survey. in Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp 3428 – 3433 (2005).
5. Badreddin, E.: Recursive Nested Behavior Control Structure for Mobile Robots, International Conference on Intelligent Autonomous Systems 2, (1989).
6. Wagner, A., Atkinson, C., Badreddin, E.: Towards a Practical, Unified Dependability Measure for Dynamic Systems, in Proc. of the International Workshop on the Design of Dependable Critical Systems, Hamburg, Germany, Sept. 15, (2009).
7. Rüdiger, J., Wagner A., Badreddin E.: Behavior Based Definition of Dependability for Autonomous Mobile Systems, in Proc. of the European Control Conference 2007, Kos, Greece, July 2-5, 2007, WeD11.4, (2007).
8. Abdel-Geliel, M., Badreddin, E., Gambier, A.: Application of Dynamic Safety Margin in Robust Fault Detection and Fault Tolerant Control, IEEE International conference on Control Applications, October 4-6, (2006).

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# Quantifying Safety in Software Architectural Designs

Atef Mohamed and Mohammad Zulkernine

School of Computing
Queen's University, Kingston
Ontario, Canada K7L 3N6
{atef, mzulker}@cs.queensu.ca

**Abstract.** Incorporating safety in the software architectural design decisions is important for the successful applications in safety-critical systems. However, most of the existing software design rationales do not consider the quantitative aspect of the software architectures with respect to safety. As a result, alternative architectures cannot be compared adequately with respect to safety. In this paper, we present an analytical approach for quantifying safety in software architectural designs. We use the concept of architectural service routes to quantify system safety in terms of software architectural attributes. We show how to make appropriate architectural design decisions based on their impacts on safety. We compare different example architectures with respect to system safety.

**Key words:** Software architecture, architectural design decisions, and system safety.

## 1 Introduction

Appropriate architectural design decisions are important for achieving quality attributes in software intensive systems. These decisions are to be taken in the early design stages and their impacts are carried out among the later development stages. *System safety* is the absence of catastrophic consequences on the system user(s) and the environment [1]. In safety-critical systems, failure types differ with respect to their criticalities (catastrophic impacts) [5]. For example, a traffic light system is highly critical to content failure (incorrect service), where the traffic lights are green in all directions. On the other hand, it is less critical to silent failures (service stopping), where all lights are turned off. An aircraft control system is more critical to silent failures than a production line control system's criticality to the same failures. Unfortunately, safety has not been sufficiently addressed at the software design level, and the quantitative impacts of software architectures on safety have not been explicitly considered in the existing software architectural design methodologies. As a result, existing architectural strategies fail to sufficiently incorporate the rationale behind the adoption of alternative architectural mechanisms with respect to their impacts on system safety [13].

Few techniques consider software architectural design decisions with respect to their impacts on safety. These techniques mainly provide a set of requirements to achieve system safety [13, 10, 3] or provide safety analysis mechanisms [5, 12]. Weihang Wu *et al.* [13] introduce some software architectural design tactics to consider safety in software architectures. The approach extends existing software architecture design tactics to consider system safety through the appropriate elicitation, organization, and documentation. Swarup *et al.* [10] propose a framework for achieving system safety through system hazard analysis, completeness of requirements, identification of software-related safety critical requirements, safety-constraints based design, runtime issues management, and safety-critical testing. Hill *et al.* [3] identify a number of safety requirements that must be possessed by a system or system component. These requirements are identifiability, stability, completeness, clarity, validity, and feasibility. Leveson *et al.* [5] and Tribble *et al.* [12] provide safety analysis based on architectural designs using Fault Tree Analysis (FTA) mechanism. FTA allows the detection of unsafe computational states and consequently, it prevents safety critical failures. However, current techniques disregard the quantitative evaluation of safety in software architectures that can incorporate system safety through the appropriate selection of the architectural design decisions.

In this paper, we present an analytical approach for quantifying safety of software architectural designs. We evaluate system safety in terms of software architectural attributes using the concept of Architectural service routes (ASRs) [8]. The concept of Architectural service routes allows quantifying architectural quality attributes by viewing a software architecture as a set of components and a set of service routes connecting them. We provide an architectural design decision approach for selecting the appropriate architecture based on its impact on safety. Finally, we compare three different example architectures based on their impacts on safety. We use "Make To Order" manufacturing planning process in our example architectures.

## 2 Preliminaries

*Software architecture* of a system is the structure, which comprises software components, the externally visible properties of those components, and the relationships among them [2]. *A component* is a unit of composition with contractually specified interfaces, explicit context dependencies only, and no persistent state. A *component interface* is a mean by which a component connects to another component [11]. A component has one or more *provided* and/or *required* interfaces [9]. *A component service* is a facility that a component provides to, or requires from other components as specified in the formal contracts with these components. Software failures are classified from *failure domain* viewpoint as content, silent, early service delivery, performance, halt, and erratic failures. [1, 6]. We denote the set of all failure types by $T$. *Failure criticality* is the estimated degree of catastrophic impact by the failure occurrence.

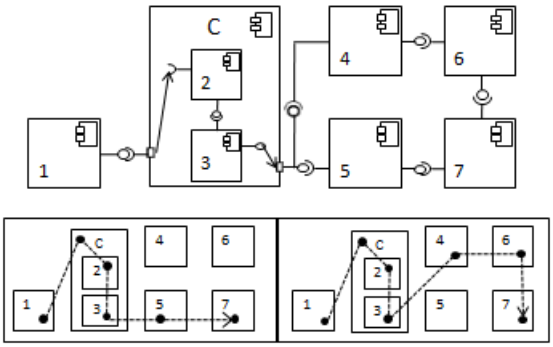



**Fig. 1.** Architectural service routes of an example architecture

*An ASR* is a sequence of components that are connected using "provided" or "required" interfaces [8]. Fig. 1 shows some ASRs of an example architecture in UML 2.0. Component 2 provides service to component 3. Component 3, on the other hand, provides services to both components 4 and 5. Therefore, component 2 provides its service to components 4 and 5 indirectly through component 3. In the bottom part of Fig. 1, we show two example ASRs between components 1 and 7 of the provided component diagram. The sequences of components are $(1, 2, 3, 5, 7)$ and $(1, 2, 3, 4, 6, 7)$ for the left and the right ASR, respectively.

Any two components $x$ and $y$ can have 0 or more ASRs. In Fig. 1, components 4 and 5 have 0 ASR, components 2 and 6 have 1 ASR: (2,3,4,6), and components 3 and 7 have 2 ASRs: (3,4,6,7) and (3,5,7). We refer to the set of ASRs from $x$ to $y$ as $\Psi^{xy}$, and we denote an ASR in this set as $\psi_k^{xy}$, where $k$ is the index of the *k-th* ASR in $\Psi^{xy}$. The length of an ASR $\psi_k^{xy}$ (referred as $L_k^{xy}$) is the number of components in it. In Fig. 1, $L_1^{2,6} = 4$, $L_1^{3,5} = 2$, and $L_1^{4,5} = 0$. $|\Psi^{xy}|$ denotes the number of ASRs from component $x$ to component $y$ *e.g.*, $|\Psi^{3,6}| = 1$ and $|\Psi^{3,7}| = 2$.

## 3 Evaluating system safety

To derive system safety in terms of architectural attributes, we exploit the results of the failure propagation analysis using ASRs [8]. Failure propagation indicates the probability that a failure propagates through system components. The quantitative evaluation, parameters, and assumptions are described in the rest of this section.

From the combinatorial viewpoint, *system safety* is the non-occurrence probability of failures that can lead to a mishap or hazard, whether or not the intended function is performed [4]. Therefore, system safety $S$ is expressed as $\prod_{f \in T}(1 - \lambda^f p^f)$, where $p^f$ is the probability of occurrence of system failure $f$, and $\lambda^f$ is the criticality of failure $f$ [7]. Failure criticality can be estimated based on expert opinion or design documents.

By considering failure propagation in software architectures, a system failure occurs when a component failure is propagated along an ASR to one of the output

interface components. Given that, we can rewrite the safety equation as follows, $S = \prod_{i=1}^{I} \prod_{f \in T} (1 - \lambda^f p_i^f)$, where $I$ is the number of system output interface components, and $p_i^f$ is the probability of occurrence of failure $f \in T$ at the system output interface component $i$. We can also replace $p_i^f$ by $\sum_{j=1}^{J} p_j^f P_{ji}^f$, where $J$ is the number of system components, $p_j^f$ is the failure probability of component $j$, and $P_{ji}^f$ is the probability of failure propagation from component $j$ to interface component $i$. *I.e.*,

$$S = \prod_{i=1}^{I} \prod_{f \in T} (1 - \lambda^f \sum_{j=1}^{J} p_j^f P_{ji}^f) \tag{1}$$

Eq. 1 evaluates system safety based on failure propagation and failure criticality. Failure propagation from any component $j$ to interface component $i$ is calculated in [8] as follows.

$$P_{ji}^f = \sum_{k=1}^{|\Psi^{ji}|} \beta^{2L_k^{ji}|T|} \tag{2}$$

where $\beta$ is a any value from 0 to 1, which expresses component failure probabilities of system components. $|T|$ is the number of failure types considered in the evaluation. (*e.g.*, $|T| = 3$ to consider content, silent, and performance failures). By substituting from Eq. 2 into Eq. 1, we get the system safety as follows.

$$S = \prod_{i=1}^{I} \prod_{f \in T} \left(1 - \lambda^f \sum_{j=1}^{J} \left(p_j^f \sum_{k=1}^{|\Psi^{ji}|} \beta^{2L_k^{ji}|T|}\right)\right) \tag{3}$$

Eq. 3 shows system safety in terms of the software architectural attributes and failure criticalities.

## 4 Architectural design decision for incorporating safety

Software designers of safety critical systems often need to select an architecture from a set of alternative architectures based on their impacts on safety. The propagation of safety-critical failures among these architectures directly impacts system safety based on the ASR attributes as shown in the previous sections. In this section, we show how to consider the quantitative evaluation of system safety in the architectural design decisions.

We provide an algorithm for evaluating system safety and selecting the appropriate architecture based on the ASR attributes among system components. Algorithm 1 provides one of the following decisions to choose between the two architectures $A$ and $A'$. *SELECT-A* indicates that architecture $A$ is selected, while *SELECT-A'* represents the selection of architecture of $A'$. *SELECT-EITHER* means that both architectures have equal impact on system safety.

The algorithm allows considering specific failure types in the architectural design decision (Line 1). For example, by considering only content failures, the

**Algorithm 1** Architectural design decision based on safety

**Input**: Architectural attribute values.
**Output**: Selected architecture.

01. Identify the set of failure types $T$ for comparing $A$ and $A'$
02. Identify the failure criticality $\lambda^f$ for each $f \in T$
03. FOR each component $j$ of architecture $A$ DO
04. FOR each output interface component $i$ of $A$ DO
05. Identify the set of ASRs between $j$ and $i$;
06. END FOR
07. END FOR
08. FOR each component $j'$ of architecture $A'$ DO
09. FOR each output interface component $i'$ of $A'$ DO
10. Identify the set of ASRs between $j'$ and $i'$;
11. END FOR
12. END FOR
13. Calculate safety for architecture $A$ and $A'$ using Eq. 3;
14. IF (safety of $A >$ safety of $A'$) THEN RETURN *SELECT-A*;
15. IF (safety of $A <$ safety of $A'$) THEN RETURN *SELECT-A'*;
16. IF (safety of $A =$ safety of $A'$) THEN RETURN *SELECT-EITHER*;

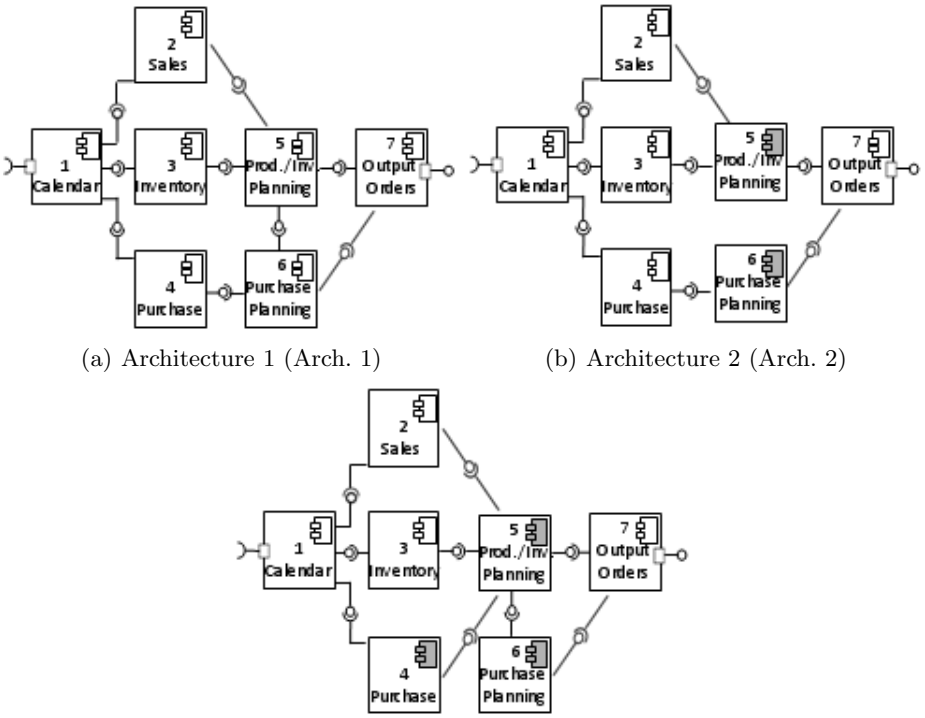
approach will compare software architectures based on data corruption among their component interactions. By considering early service delivery and late service delivery failures, the architectures will be compared based on their performances. Algorithm 1 selects the architecture that has the higher safety value quantitatively. In Line 2, the failure criticalities are identified for the failure types in the set $T$. These failure criticalities can be identified based on expert opinion or design documents. Lines 03-07 calculate the failure propagation probabilities between each pair of components for architectures $A$. Similarly, Lines 08-12 calculate the failure propagation probabilities for architectures $A'$. Line 13 calculates the safety of architecture $A$ and $A'$. Based on the quantified safety of $A$ and $A'$, Lines 14-16 select the architecture with the higher safety.

## 5 Case study: comparing safety of example architectures

We use the example of the "Make To Order" (MTO) production planning process of manufacturing systems to explain the proposed technique for comparing different architectures. In MTO, products are manufactured after a confirmed sales order is received for them. We present three different example architectures for this process in Fig. 2. We evaluate the safety of these architectures based on their ASR attributes. Each of the architectures in Fig. 2 uses 7 components, numbered from 1 to 7. Component 1 is an input interface component, in which the user inputs the production planning intervals. It passes the planning intervals to three other components (sales, inventory, and purchase orders) after checking the manufacturing schedule according to the calendar. Component 2 and component 3 deliver the corresponding sales orders and item inventory to the production and inventory planning component 5. Component 4 delivers the purchase orders to the purchase planning component 6. Component 5 also delivers the planned inventory requirements to component 6. Finally, both component 5

and component 6 deliver their outputs to component 7 to create inventory outbound, purchase orders, and planned production orders.

The three architectures differ slightly in the interface with respect to the shaded components. Unlike Fig. 2(a), Fig. 2(b) does not have a connector between components 5 and 6. Fig. 2(c) differs from Fig. 2(a) in that the connector between components 4 and 6 is removed, and another connector between components 4 and 5 is added. Regardless of the functional advantages or disadvantages of these changes, we study these three architectures to see their impacts on the overall system safety. We consider three failures (content, silent, and performance failures), *i.e.*, $|T| = 3$. In the computation of safety, we assume $\beta = 0.7$, since smaller values may result in more approximations and less preciseness. For simplicity, we choose $p_j^f = 0.001$ for all components and $\lambda^f = 0.5$ for all types of failures. We use Eq. 3 to obtain the system safety.




(a) Architecture 1 (Arch. 1)

(b) Architecture 2 (Arch. 2)

(c) Architecture 3 (Arch. 3)

**Fig. 2.** Different example architectures of "Make To Order" production planning.

Here, we show how to obtain the ASR attributes using Arch. 1 as an example. We also show how to use these attributes to calculate system safety. Since component 1 is an input interface component and component 7 is an output interface component, there is no interface connection from any component to component 1 or from component 7 to any other component.

Table 1.*a*, 1.*b*, and 1.*c* correspond to Arch 1, 2, and 3 respectively. In each

table, rows represent component numbers from 1 to 6 and columns represent component numbers from 2 to 7. The table cells represent the ASR attributes among components in the form ($N_1$x$M_1$, $N_2$x$M_2$,...), where $N_i$x$M_i$ means: there exist $N_i$ ASRs of length $M_i$. For example, a cell (row= 1, column= 5) of Arch. 1 has the value 2x3 since $\Psi^{1,5} = 2$ and both $L_1^{1,5}, L_2^{1,5} = 3$. Similarly, cell (row= 1, column= 7) of Arch. 1 represents $\Psi^{1,7}$ and has the value of 3x4, 2x5. $\Psi^{1,7}$ includes 5 ASRs as follows, $\Psi^{1,7} = \{(1,2,5,7), (1,3,5,7), (1,4,6,7), (1,2,5,6,7),$ and $(1,3,5,6,7)\}$ for $\{\psi_1^{1,7}, \psi_2^{1,7}, \psi_3^{1,7}, \psi_4^{1,7},$ and $\psi_5^{1,7}\}$, respectively. The lengths of the ASRs are $4, 4, 4, 5,$ and 5, respectively. By considering the ASR attributes in Table 1 and the previously mentioned values of $p_j^f$, $\beta$, $\lambda^f$, and $|T|$ in Eq. 3, we get, $S = 0.998887872$ for Arch. 1 where $S \in [0, 1]$.

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | 1x2 | 1x2 | 1x2 | 2x3 | 1x3,2x4 | 3x4,2x5 |
| 2 | 0 | 0 | 0 | 1x2 | 1x3 | 1x3,1x4 |
| 3 | 0 | 0 | 0 | 1x2 | 1x3 | 1x3,1x4 |
| 4 | 0 | 0 | 0 | 0 | 1x2 | 1x3 |
| 5 | 0 | 0 | 0 | 0 | 1x2 | 1x2,1x3 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1x2 |

(a) Arch. 1

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | 1x2 | 1x2 | 1x2 | 2x3 | 1x3 | 3x4 |
| 2 | 0 | 0 | 0 | 1x2 | 0 | 1x3 |
| 3 | 0 | 0 | 0 | 1x2 | 0 | 1x3 |
| 4 | 0 | 0 | 0 | 0 | 1x2 | 1x3 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1x2 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1x2 |

(b) Arch. 2

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | 1x2 | 1x2 | 1x2 | 3x3 | 3x4 | 3x4,3x5 |
| 2 | 0 | 0 | 0 | 1x2 | 1x3 | 1x3,1x4 |
| 3 | 0 | 0 | 0 | 1x2 | 1x3 | 1x3,1x4 |
| 4 | 0 | 0 | 0 | 1x2 | 1x3 | 1x3,1x4 |
| 5 | 0 | 0 | 0 | 0 | 1x2 | 1x2,1x3 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1x2 |

(c) Arch. 3

Table 1: ASR attributes of architecture 1, 2, and 3.

Similarly, based on the ASR attributes of Arch. 2 provided in Table 1, the system safety for Arch. 2 is 0.998908816. Comparing the safety values of Arch. 2 and Arch. 1, we can conclude that the Arch. 2 is safer than Arch. 1. This safety gain in Arch. 2 is due to the decrease in the number of ASRs from the system components in general to the output interface component. For example, $|\Psi^{1,7}| = 3$ in Arch. 2, while $|\Psi^{1,7}| = 5$ in Arch. 1. The decrease in the number of ASRs between two components decreases the propagation probabilities and consequently increases the system safety. In Arch. 3, we have increased the number of ASRs (*e.g.*, $|\Psi^{1,7}| = 6$ instead of 5 for Arch. 1) and the lengths of the shortest ASRs (*e.g.*, $L_S^{4,6} = 2$ instead of 1 for Arch. 1). According to our analysis, these changes should decrease the system safety.

Based on the ASR attributes of Arch. 3 shown in Table 1, the safety is calculated as $S = 0.998887773$. Comparing Arch. 3 and Arch. 1, the safety is lower for Arch. 3. The lower safety in Arch. 3 is due to the increase in the number of ASRs among system components. Comparing Arch. 3 and Arch. 2, the safety is lower for Arch. 3. This loss of safety is also due to the increase in the number of ASRs among system components.

## 6 Summary and future work

Safety has not been sufficiently addressed and the quantitative impacts of software architectures on this quality attribute have not been explicitly considered in the existing software architectural design methodologies. As a result, existing architectural strategies fail to sufficiently identify the rationale behind the

adoption of alternative architectural mechanisms with respect to safety. In this paper, we present an analytical approach for quantifying safety in software architectural designs. We evaluate system safety in terms of software architectural attributes using the concept of ASRs. Finally, we provide an architectural design decision approach for selecting the appropriate architecture based on their impacts on safety-critical failure propagation among system components. The main contribution of this work is to provide a quantitative evaluation of system safety based on software architecture in an early design stage of software system development. In our future work, we plan to estimate the criticality of a component based on its location and connectivity in an architecture. This will help to identify the components that are critical to system safety.

## References

1. A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing", *IEEE Transactions on Dependable and Secure Computing*, Mar 2004, Vol: 1, pp. 11- 33.
2. L. Bass, P. Clements, and R. Kazman, "Software Architecture in Practice". 2-*nd* eddition. 2003: Addison-Wesley.
3. J. Hill and D. Victor, "The Product Engineering Class in the Software Safety Risk Taxonomy for Building Safety-Critical Systems", *Proc. of the 19th Australian Conference on Software Engineering*, 2008, pp. 617-626.
4. N.G. Leveson, "Software safety: why, what, and how", *ACM Computing Surveys (CSUR) archive*, Jun 1986, Vol 18, pp. 125-163.
5. N.G. Leveson and P.R. Harvey, "Analyzing Software Safety", *IEEE Trans. on Software Engineering*, Sep 1983, Vol SE-9, NO. 5, pp. 569-579.
6. B. Littlewood and L. Strigini, "Software reliability and dependability: a roadmap", *Proc. of the 22nd IEEE International Conference on Software Engineering on the Future of Software Engineering (ICSE'00)*, Limerick, Ireland, 2000, pp. 175-188.
7. A. Mohamed and M. Zulkernine, "Improving Reliability and Safety by Trading off Software Failure Criticalities", *Proc. of the 10th IEEE International Symposium on High Assurance System Engineering*. Nov 2007, Dallas, Texas, pp. 267-274.
8. A. Mohamed and M. Zulkernine, "On Failure Propagation in Component-Based Software Systems", *Proceedings of the 8th IEEE International Conference on Quality Software*, IEEE CS Press, Oxford, UK, 2008, Pg: 402-411.
9. Object Management Group, "OMG Unified Modeling Language (OMG UML)", Superstructure, Version 2.1.2, *OMG Available Specification without Change Bars*, formal/2007-02-05, Nov 2007.
10. M.B. Swarup and P.S. Ramaiah, "An Approach To Modeling Software Safety", *Proc. of the 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 2008, pp. 800-806.
11. C. Szyperski, "Component software: beyond object-oriented programming", *Addison-Wesley*, 1998, ISBN 0-201-17888-5.
12. A.C. Tribble and S.P. Miller, "Software Intensive Systems Safety Analysis", *IEEE A&E Systems Magazine*, Oct 2004, pp. 21-26.
13. W. Weihang and T. Kelly, "Safety tactics for software architecture design", *Proceedings of the 28th Annual International Conference on Computer Software and Applications.*, York Univ., UK, Sep 2004, pp. 368-375.

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# The Role of Task and Situational Characteristics on the Dependability of Human-Technology Interaction

Meike Jipp, Christian Bartolein, Essameddin Badreddin

Automation Laboratory, University of Heidelberg, Germany
{meike.jipp, christian.bartolein, badreddin}@ziti.uni-heidelberg.de

**Abstract.** While the impact of "human error" on failures of complex human-technology systems has widely been demonstrated and accepted, the relevance of situational and task-related characteristics on human performance has not yet been considered sufficiently. For this purpose and on the example of electrically powered wheelchair control this paper analyzes the effects of situational characteristics (e.g., turns to the left/right in the backward/forward driving mode) on the impact of fine motor abilities on human performance. A study with 23 participants is described in the paper, during which relevant data such as the subjects' precision and aiming capacity, the number of collisions caused while driving as an indicator for human performance, and the situational characteristics were measured. The data analyses demonstrate an influence of especially the number of turns driven to the right in the backward mode on the impact of the precision ability on the number of safety-critical collisions. The results highlight the necessity not only to develop a wheelchair system which is adaptable to the user's fine motor abilities, but also to the situational characteristics in order to increase the dependability of the human-technology system at hand.

**Keywords:** human-technology interaction, powered wheelchair control, fine motor abilities, adaptive automation systems, situational characteristics

## 1 Motivation and State of the Art

Statistics and analyses of failures of human-technology systems demonstrate the impact and most importantly the exponential rise of the so-called *human error*, classically categorized as either an error of commission or an error of omission. According to Hollnagel [1], the human operator contributed to about 20% of system errors in 1960. In 1990, this same percentage has risen up to 90% (cf. [2]). A number of reasons are discussed in the literature – covering the increasing complexity of the technical systems and the resulting incapability of the human operator to maintain a high level of situation/mode awareness, incorrect mental models of the technical system at hand, a loss of manual skills, etc. (cf. [3], [4], [5]).

In order to improve these statistics, the field of human reliability analyses has emerged, which first generation methods (e.g., Technique for Human Error Rate Prediction, THERP, [6]) aimed (1) at functionally decomposing human tasks, (2) at

identifying performance shaping factors (e.g., cognitive abilities, fatigue, illness, experience/qualification, weather conditions, automation design), which are expected to impact the implementation of these (human) tasks, and (3) at mathematically combining this information to yield a probability number reflecting the likelihood of a human error in advance. The second generation methods criticized these first generation methods due to their roots in the field of probabilistic risk assessment, which ignored the cognitive characteristics of the human operator (cf. [7]). An example for a second generation method is the Cognitive Reliability Error Analysis Method (CREAM) ([7]), which is based on a cognitive model of human performance. Due to this theoretical foundation, the method can either be used post hoc for accident analyses, but also for a priori performance predictions, which allow developing reasoning algorithms impeding the human error by replacing the human function with appropriate automation.

## 2 Problem Formulation

While already the term *human error* implies that the human being itself plays a major role, it is often not considered sufficiently that human behavior is a function of the person **and** his/her environment. This is reflected in the, in the meantime, well-established behavior equation of Kurt Lewin [8]. While the "person-component" and its impact has been tested in the field of human-technology interaction (cf. [9]), the relevance especially of task and situational characteristics on the relationship between human characteristics and performance will be analyzed in this paper on the example of a safety-critical system, i.e., an electrically powered wheelchair for people with severe disabilities.

## 3 Solution Approach

In order to provide evidence for the impact of task and situational characteristics on the influence of human abilities on their performance, a study has been conducted, which is in the following thoroughly described and discussed.

### 3.1 Description of the Course of the Study

In order to collect data on the occurrence of safety-critical collisions, the study's participants were first asked to drive through a standardized course with 14 sections in a realistic office environment. Therefore, an electrically powered wheelchair was used, which is commercially available from the company Otto Bock Healthcare GmbH (type B600). This wheelchair has been equipped with additional hard- and software in order to be able to record the required data, but also to provide additional assistive functionality such as collision avoidance, which has, however, for this study been switched off. The wheelchair, as it was applied here, has thoroughly been described in [10]. While driving data such as the route or the time required for

reaching a defined goal position as well as the number of caused collisions, were recorded.

The course, which the participants had to drive through, was designed such that a number of supposedly critical behaviors (e.g., turning on the spot; driving around corners) were evoked in order to be able to relate such task/situational characteristics with human abilities and their performance.

In a second step, the participants' fine motor abilities were diagnosed with the "Motor Performance Test" of Neuwirth and Benesch [11], which is necessary in order to answer the stated research question.

Last, the participants were asked to fill in a biographical questionnaire assessing data for example on the age of the participants, their gender, field of study, etc.

### 3.2 Description of the Sample

Out of practical considerations, the convenience sample consisted of 23 students of the Universities of Heidelberg and Mannheim (Germany). The students were not disabled. In order to be able to control e.g. skill acquisition effects, the participants had unlimited time available to practice maneuvering with the wheelchair in the environment, in which the actual data recording took place.

The majority of the participants were Bachelor students enrolled in psychology ($n$ = 20), while $n$ = 3 were Master students in computer engineering. In addition, 12 participants were female, 11 were male.

### 3.3 Data Analyses

In order to relate the characteristics of each course section with the number of collisions and the participants' fine motor abilities, we first of all identified the critical situational characteristics of the course by counting especially the number of turns which needed to be driven in the forward mode to the right and to the left, the number of times, a participant had to drive straight backward, the number of times, the participant had to drive a turn to the right/left in the backward mode and the number of times the participant had to turn on the sport to the right and to the left. In order to demonstrate that there were no sincere dependencies between these variables, their correlations were calculated (see Tab. 1).

As Tab. 1 shows, these correlations vary between $r$ = 0.339 ($p > 0.05$) and $r$ = -0.552 ($p < 0.05$). The latter correlation is the only one, which has reached an acceptable level of significance and reflects the fact that, if a course section contained turns to the right (to be driven in the forward mode), less turns to the left (also to be driven in the forward mode) had to be made in order to achieve the current goal position. Hence, despite this correlation, there were no significant relationships between the different task characteristics in the course.

In a second step, inferential statistics were applied in order to test whether these situational characteristics have an influence on the relationship between the impact of the fine motor abilities on the number of collisions caused while driving.

For these purpose, we calculated univariate analyses of variance with the described situational characteristics as independent variables. As dependent variable, we used the impact of (1) the precision ability and (2) the aiming capacity on the number of collisions (see also [12], [13]). This impact can statistically be described as an effect size [14]. The results of the univariate analyses of variance regarding the precision ability are summarized in Tab. 2.

**Table 1.** Correlations between the task and situational characteristics of the course

| | Number of turns to the right, forward mode | Number of turns to the left, forward mode | Backward, straight ahead | Number of turns to the right, backward mode | Number of turns to the left, backward mode | Turning on the spot to the right | Turning on the spot to the left |
|---|---|---|---|---|---|---|---|
| Number of turns to the right, forward mode | - | | | | | | |
| Number of turns to the left, forward mode | -0.552* | - | | | | | |
| Backward, straight ahead | 0.077 | -0.439 | - | | | | |
| Number of turns to the right, backward mode | -0.372 | 0.025 | 0.240 | - | | | |
| Number of turns to the left, backward mode | -0.025 | -0.322 | 0.240 | -0.077 | - | | |
| Turning on the spot to the right | 0.057 | 0.339 | -0.228 | -0.439 | -0.439 | - | |
| Turning on the spot to the left | -0.025 | 0.025 | 0.240 | -0.077 | -0.077 | -0.439 | - |

* $p < 0.05$

As Tab. 2 demonstrates, there is a highly significant effect ($F(1, 12) = 103{,}14$, $p = 0.00$, $f^2 = 0.90$) of the number of turns to the right driven in the backward mode on the impact of the precision ability on the number of collisions caused while driving. To visualize this effect, a line plot is displayed in Fig. 1, which shows that the

greater the number of turns driven to the right in the backward mode, the greater the relationship between the number of caused collisions and the precision ability.

**Table 2.** Results of the univariate analyses of variance with the relationship between the precision ability and the number of collisions as a dependent variable

| Independent Variable | Value of the test statistic *F* | Probability *p* | Effect size $f^2$ |
|---|---|---|---|
| Number of turns to the right, forward mode | $F(1, 12) = 0.95$ | 0.38 | 0.07 |
| Number of turns to the left, forward mode | $F(1, 12) = 0.08$ | 0.79 | 0.01 |
| Number of times driven backward, straight ahead | $F(1, 12) = 0.21$ | 0.66 | 0.02 |
| Number of turns to the right, backward mode | $F(1, 12) = 103.14$ | 0.00** | 0.90 |
| Number of turns to the left, backward mode | $F(1, 12) = 0.12$ | 0.74 | 0.01 |
| Turning to the right on the spot | $F(1, 12) = 1.89$ | 0.19 | 0.14 |
| Turning to the left on the spot | $F(1, 12) = 0.12$ | 0.74 | 0.01 |

** $p < 0.01$

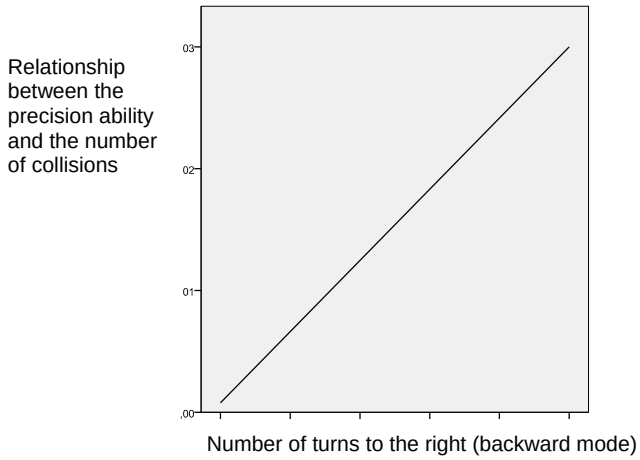



**Fig. 1.** Line plot of the relationship between the effect of the precision ability on the number of collisions while driving through the course and the number of turns to the right.

In a next step, we analyzed the impact of the situational characteristics of the course sections on the relationship of the aiming capacity and the number of collisions. Again, we calculated univariate analyses of variance with the situational characteristics as independent measures and the relationship (i.e., the effect sizes) between the aiming capacity and the caused collisions as a dependent variable. The results are given in Tab. 3.

As Tab. 3 demonstrates and in contrast to the results introduced before, no significant effects with $p < 0.05$ have been found. Hence, at least these results give the impression that the chosen situational characteristics do not influence the impact of the aiming capacity on the number of collisions. However, it is to be considered that the sample size was relatively small. As the effect sizes, which are also displayed in Tab. 3, demonstrate, there are effects, which partially have reached a medium-size according to Cohen [14]. Due to the low power of the study at hand, these effect sizes might not have reached an appropriate level of significance.

**Table 3.** Results of the univariate analyses of variance with the relationship between the aiming capacity and the number of collisions as a dependent variable

| Independent Variable | Value of the test statistic $F$ | Probability $p$ | Effect size $f^2$ |
|---|---|---|---|
| Number of turns to the right, forward mode | $F(1, 12) = 1.47$ | 0.25 | 0.11 |
| Number of turns to the left, forward mode | $F(1, 12) = 1.31$ | 0.28 | 0.10 |
| Number of times driven backward, straight ahead | $F(1, 12) = 0.92$ | 0.36 | 0.07 |
| Number of turns to the right, backward mode | $F(1, 12) = 0.03$ | 0.87 | 0.00 |
| Number of turns to the left, backward mode | $F(1, 12) = 0.11$ | 0.75 | 0.01 |
| Turning to the right on the spot | $F(1, 12) = 0.44$ | 0.52 | 0.04 |
| Turning to the left on the spot | $F(1, 12) = 0.11$ | 0.75 | 0.01 |

## 4 Discussion, Conclusions, and Future Work

Summarizing, this paper introduces the necessity to consider not only the characteristics of the human operator/user, but also task- and situation-related factors, which influence the relationship between the human operator and his/her performance. In order to demonstrate this relationship, a study has been conducted, during which participants drove through a course with an electrically powered wheelchair being one example of a safety-critical system. The course was defined such that a number of presumably critical situations occurred. The participants' collisions with objects in the environment were measured. In addition, the participants' fine motor skills were administered. In order to answer the stated research question, inferential statistics with the resulting data set were applied. More specifically, univariate analyses of variance demonstrated that the characteristics of the course sections impact the relationship between the precision ability and the number of collisions while driving: The turns which needed to be driven in a backward mode to the right side require a higher level of precision in order to avoid collisions when compared to turns which need to be driven to the left. Other effects have not reached an appropriate level of significance. This could be due to the low sample size, the inexistence of this effect or a high correlation between the situational

characteristics. However, the latter reason can be rejected, as the analysis of the correlational patterns has shown that only minor relationships existed between the occurrences of situational characteristics.

In a next step, it will be aimed at collecting additional data in order to check whether the in this study insignificant medium-sized effects actually exist. In the long run, methods will be developed, which enable a complex computer system to judge on the complexity of a future action and change its level of autonomy accordingly, such that the dependability of safety-critical human-technology systems increases.

## References

1. Hollnagel, E.: Human reliability analysis: Context and control. London: Academic Press, 1993.
2. Swain, A. D.: Human reliability analysis: Need, status, trends, and limitations. Journal of Reliability Engineering and System Safety, 29, 301-313, 1990.
3. Endsley, M. R., Kiris, E. O.: The out-of-the-loop performance problem and level of control in automation. Human Factors, 37, 381-394, 1995.
4. Parasuraman, R., Mouloua, M., Molloy, R., Hilburn, B.: Training and adaptive automation II: Adaptive manual training (Technical Report CSL-N92-2). Washington, DC: Cognitive Science Laboratory, Catholic University of America, 1992.
5. Parasuraman, R., Riley, V. A.: Humans and automation: Use, misuse, disuse, abuse, Human Factors, 39, 230-253, 1997.
6. Swain, A. D., Guttman, H. E.: Handbook of human reliability analysis with reference to the nuclear power plant application, Washington DC: U.S. Nuclear Regulatory Commission, 2-7, 1983.
7. Hollnagel, E.: Cognitive reliability and error analysis method. Oxford: Elsevier Science Ltd, 1998.
8. Lewin, K.: Principles of topological psychology. USA, McGraw-Hill, 1936.
9. Jipp, M., Pott, P., Wagner, A., Badreddin, E., Wittmann, W. W.: Skill acquisition process of a robot-based and a traditional spine surgery. Proceedings of the International Conference on Informatics in Control, Automation, and Robotics, 1(2), 56-63, 2004.
10. Bartolein, C., Wagner, A., Jipp, M., & Badreddin, E.: Multilevel intention estimation for wheelchair control. Proceedings of the European Control Conference 2007, 1, 5463-5470, 2007.
11. Neuwirth, W., Benesch, M.: Motorische Leistungsserie, Schuhfried, Mölding. 2004.
12. Jipp, M., Bartolein, C., & Badreddin, E.: Predictive validity of wheelchair driving behavior for fine motor abilities: Definition of input variables for an adaptive wheelchair system. Accepted for Publication at the IEEE International Conference on Systems, Man, and Cybernetics, 2009.
13. Jipp, M., Bartolein, C., Wagner, A., & Badreddin, E.: The impact of individual differences in fine motor abilities on wheelchair control behavior and especially on safety-critical collisions with objects in the surroundings. Accepted for publication for the Workshop on the Design of Dependable Critical Systems: "Hardware, Software, and Human Factors in Dependable System Design" in the Framework of the 28th International Conference on Computer Safety, Reliability and Security, 2009.
14. Cohen, J. : Statistical power analysis for the behavioral sciences. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

# Hierarchical Hybrid Monitoring for Autonomous Systems

Leila Zouaghi, Achim Wagner, and Essam Badreddin

Automation Laboratory, University of Heidelberg, Germany
{leila.zouaghi, achim.wagner, badreddin}@ziti.uni-heidelberg.de

Safety critical computer systems such as control systems for automobile, aircraft, medical and intelligent mobile robots, are rapidly growing in complexity. This increasing complexity has made system monitoring an inevitable component of system operations and the subject of intensive study in the past few years. Several methods are used to deal with hybrid systems monitoring, which are based on multi-model numerical filters, such as the Kalman filter [1] or particle filtering methods [2], [3]. Other approaches are based on automata [4] or on Bayesian nets [5] linked to some numerical evolution models. The only existing monitoring approach based on Particle Petri net was used for the analysis of flight procedures and deals with situation monitoring [6]. We consider a general, nonlinear, distributed, complex system with hybrid (discrete/continuous) behavior, for which a monitor has to be designed. Such systems present significant challenges for monitoring and diagnosis. For a large number of states and highly nonlinear equations, the design of a monitor is clearly problematic.

Our approach gives a solution to reduce the design complexity by decomposing such a system using separate monitors for each subsystem. In this context, we have proposed a model of hierarchical hybrid monitoring for systems with so called "Recursive Nested Behavior Control" (RNBC) structure, which has been successfully employed for autonomous mobile robots [7], [8]. Since the system architecture is nested, the monitoring system is built using a nested structure. In this scheme, the monitors of a subsystem work independently using recursively the results of the monitors of the lover levels. The monitoring concept of the RNBC is shown in Fig.1.

The hybrid state estimation is performed using a particle Petri net [9] model. It allows the representation of the discrete dynamics of the system through the Petri net structure and the modeling of the continuous behavior by evolution equations. The estimator is based on the particle filtering principle and computes the expected markings of the particle Petri net. From the estimation of the marking of the Petri net inconsistent behaviors can be detected. The consistency is checked with respect to the reachable markings of the Petri net.

This work addresses the challenge of the interaction between continuous and discrete dynamics for the monitoring of autonomous systems with nested structure. The novelty of the framework is the use of the Particle Petri net for the monitoring of systems with a Recursive Nested control structure and the methodology for detecting

discrepancies between the expected and the actual behaviour of the system in such structure. The nested hybrid estimation methodology has been demonstrated on a heating control system example. The simulation results show the feasibility of the proposed design.

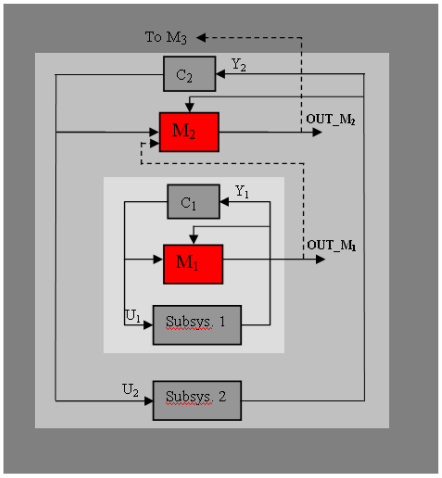



*Figure 1. Monitoring structure.*

## References

1. Veeraraghavan, H. and Papanikolopoulos, N.: "Combining multiple tracking modalities for vehicle tracking in traffic intersections". In IEEE International Conference on Robotics and Automation (ICRA), USA, 2004.
2. Koutsoukis, X., Kurien, J. and Zhao, F.: "Monitoring and Diagnosis of hybrid systems using particle filtering methods". Proc. Mathematical Theory of Networks and Systems (MTNS), 2002.
3. Verma, V.: Tractable Particle Filters for Robot Fault Diagnosis. Doctoral dissertation, Robotics Institute, Carnegie Mellon University, May, 2005.
4. Hofbaur, M. and Williams, B. : "Mode estimation of probabilistic hybrid systems". In International Workshop on Hybrid Systems, Computation and Control (HSCC), Stanford, California, USA, 2002.
5. Lerner, U., Moses, B., Scott, M., McIlraith, S., Koller, D.: Monitoring a complex physical system using a hybrid dynamic Bayes net. In: UAI'02, Edmonton, AB (2002)
6. Lesire, C., Tessier, C.: Particle Petri nets for aircraft procedure monitoring under uncertainty. In: ATPN'05, 26 th International Conference On Application and Theory of Petri Nets and Other Models of Concurrency, Miami, FL (2005)
7. Badreddin, E. "Recursive Control Structure for Mobile Robots", International Conf. on Intelligent Autonomous Systems 2 (IAS.2), Amsterdam, pp. 11-14, 1989.
8. Badreddin, E., "Recursive Behaviour-based Architecture for Mobile Robots", Robotics and Autonomous Systems, VO1.8, 1991
9. Lesire, C. et Tessier, C. : Réseaux de petri particulaires pour l'estimation symbolico-numérique. In Journées Formalisation des Activités Concurrentes (FAC), Toulouse, France, 2005b.

International Workshop on
the Design of Dependable Critical Systems
September 15, 2009, Hamburg, Germany

# Dependable System Design for Assistance Systems for Electrically Powered Wheelchairs

Christian Bartolein, Achim Wagner, Meike Jipp, Essameddin Badreddin

Automation Laboratory, University of Heidelberg, Germany
{christian.bartolein, achim.wagner, meike.jipp, badreddin}@ziti.uni-heidelberg.de

**Abstract.**
In this paper a system design approach is proposed, which is based on a user needs assessment and a flexible and adaptable architecture for dependable system integration. The feasibility of the approach is shown on the example of an assistance system for electrically powered wheelchairs. The system requirements correspond to the cognitive and motor abilities of the wheelchair users. For the wheelchair system built up based on a commercial powered wheelchair several behaviors have been realized such as collision avoidance, local navigation and path planning well known from robotic systems, which are enhanced by human-interfacing components. Furthermore, the system design will be highlighted which is based on robotic systems engineering. Due to the fundamental properties of the system architecture the resulting assistance system is inherently dependable, flexible, and adaptable. Corresponding to the current situation and the users' abilities the system changes the level of assistance during real-time operation. The resulting system behavior is evaluated using system performance and usability tests.

**Keywords:** dependability, system design, user needs assessment, requirement analysis, use cases, system architecture, evaluation

## 1 Introduction: Motivation, State of the Art, and Research Question

According to a survey of the University of Berkley, California, published in 2002 the number of computing systems used in everyday life is expected to grow at a percentage rate of 38% per annum. At the same time, the degree of complexity of these computing systems is increasing. Some specialists even warn [1] about this "nightmare of pervasive computing" due to the inability of the system designers to anticipate, design, and maintain such complex systems interacting with each other which can result in catastrophic consequences especially when dealing with safety-critical systems. To enable system designers to develop such complex systems consisting of hard- and software and to consider human factors, an appropriate system design approach is required. This system design approach should, on the one hand, offer methodologies which enable the integrated consideration of these three system components, and, which, on the other hand, supports the dependability of the overall system, thus, decreasing the possibility of a sincere system failure.

A system design approach which meets these requirements is introduced in the following sections theoretically and demonstrated exemplarily on the demonstration platform “assistance system for powered wheelchairs”.

## 2 Dependability-Centered System Design Considering Software, Hardware, and Human Factors

The dependability-centered system design approach advocated here consists of a number of steps, which are thoroughly described in the following.

### 2.1 User Needs Assessment

A user needs assessment is an evaluative study or an experiment that gives answers about the condition a system is attended to address (cf. [2]). It may also be used in order to compare or prioritize different needs which can be tackled. In order to derive these answers, different methodologies are available (for an overview cf. [3, 4]) ranging from qualitative research designs such as formative scenario analyses or future workshops to quantitative experiments. As thoroughly described in [4] each method provides important insights and has its own advantages and disadvantages, such that only a multi-method approach [5] allows deriving meaningful and valid results. While the quantitative research methods offer a high internal validity, so that a found effect can with great certainty be traced back to the experimental manipulation; they only have a low external validity, which reflects the poor generalizability of the results to other settings, other persons and other timings. This is the case as the experiments take place in a restricted laboratory environment [6]. Vice versa, the qualitative methods allow generalizing the results; however, the results can only to a limited extent be traced back to a manipulation. This is the case as other causes such as sample biases cannot be eliminated [6].

With regard to the wheelchair application the user needs assessment was realized in one study, during which about 15 participants with different types of disabilities executed a gardening task (for a more thorough description, see [7]), and in an experiment, during which about 20 healthy participants drove through a standardized course in a realistic office environment with a given electrically powered wheelchair, however, with different control methods (for a more detailed description, see [8]). In the above introduced classification, the first study reflects a qualitative research method design, as it does not contain any experimental manipulation (all participants executed the same tasks with the same tools). In addition, the participants were asked to fill in unstructured questionnaires. Hence, the study allows generalizing the results. The second data acquisition was an experiment in the classical sense, although the experimental manipulation was a within-subject manipulation and not a between-subject one. The experimental manipulation, we were interested in, is the control mode of the wheelchair. On the one hand the wheelchair could be steered with a standard joystick; on the other hand, the wheelchair was controlled with a two-switch control reflecting a speciality input control device. While a between-subject

experimental variation would have requested us to split our pool of participants and let one group execute the course with the joystick control mode; the second group would have been asked to use the two-switch control mode. Due to the small number of participants, which was available, we asked each participant to drive through the course twice – the first time with the joystick control mode, the second time with the two-switch control mode. While driving we collected data on the collisions which were evoked by the driving behavior of the participants.

The results of the study are two-fold: On the one hand, the questionnaire/qualitative data indicated that especially people with spasticities have troubles operating a standard joystick especially in acute phases. In addition, they have troubles interpreting figural information, e.g., a city map. Furthermore, people suffering especially from incomplete paralysis have deteriorating abilities which requires them to continuously adjust their wheelchair such that they can benefit from it in their everyday life. On the other hand, the quantitative data derived from the study (for a thorough description of the data analyses, see [9]) shows that the variation of the cognitive and fine motor abilities of the participants is quite large and that this variation is to a great degree predictive for behavior differences for wheelchair users.

The experiments' results (cf. [10]) demonstrate that individual differences in the fine motor abilities of the participants were highly indicative about their wheelchair behavior. This refers e.g. to the number of collisions which occurred while driving through the realistic office environment, but also to the velocities driven or to the number of input commands administered to the technical system at hand.

Hence, by applying different research methods for the user needs assessment it enables us (1) to actually trace back the found effects to the individual differences of the users and (2) to generalize this effects to other samples out of the wheelchair population. It is, thus, a thorough basis for deriving the system requirements.

### 2.2 System Requirements

The goal of this step in the dependability-centered system design approach is to derive a description of the system, which matches as many as possible of the identified user needs. In order to yield these system requirements, the process advocated is based on the ISO Norm 13407 and the socio-cognitive engineering approach. More specifically, a workshop with the design engineers should be conducted, during which the following steps need to be covered:

- specifying a design concept which does meet the needs of the potential users, e.g. by using the design ideas of potential users as an important source of inputs for the design concept
- generating a space of possible system designs, which will make the design concept more concrete by working out different ways of design ideas which will enable to achieve the set design concept
- specifying the functional and non-functional aspects of the system (including the technical specifications) – the functional and non-functional aspects of the system at hand will be worked out for all possible system designs and the one chosen, which, from a technical point of view, yields an optimal solution to the perceived problem situation of the people in need and their task model

- yielding feedback on the functional and non-functional aspects of the envisioned system on the basis of qualitative research methods

In order to derive these system requirements on the example of the assistance system for electrically powered wheelchairs, the results of the user needs assessment were thoroughly presented during a workshop and potential design ideas discussed and reviewed. One potential solution was reflected in a wheelchair which offers high assistive functionality. If, e.g., global navigation and collision avoidance was provided to the wheelchair user, this should have the potential (1) to significantly reduce the possibility of the occurrence of safety-critical situations, as it reduces the impact of the user's input on the wheelchair behavior, and (2) to improve the disadvantages of today's wheelchair control when applying speciality input devices (e.g., reduce the number of input commands, reduce the time required to reach a goal position, optimize the distances to reach an object, etc.). This design idea was then presented to stakeholders. While the actual users liked the idea of a highly autonomous wheelchair, critics came from nurses and physicians, who feared skill degradation. Due to these issues, a nearly autonomous wheelchair as a potential design solution was rejected and another design worked out, which has actually reached positive feedback from all stakeholders and which is described in the following:

Due to the great variability of abilities within and between potential users and their severe impact on the occurrence of safety-critical situations and human performance differences, an assistance system for electrically powered wheelchairs should first of all offer different levels of autonomy, which provide different levels of assistive functionality to the user. Second, these levels of autonomy should automatically be adaptive to the current ability level of its user (cf. [10]). The automatic adaptation is crucial to offer as much support as necessary in this moment, but not as much support as possible. In addition and especially due to the problems related to the interpretation of figural information for some users with specific disabilities, not only the level of autonomy should be adaptive, but also the content representation on the interface. Besides these functional requirements, non-functional requirements with regard to the dependability and the maintainability of the overall human-technology system were set.

### 2.3 Use Cases

In order to guarantee the common understanding of the envisioned system, use cases need to be worked out in a next step, which describe how a typical user might use the system at hand (cf. [11]).

On the example of the assistance system for electrically powered wheelchairs, the following use case has been worked out:

A wheelchair user with spastics, which are currently on a low level, uses the - in the previous section - described adaptive assistance system. After the first interactions with the system, the assistance system knows about the user's current good ability level and activates the low assistance functionality mode. This low assistance functionality mode uses a collision avoidance behavior on the basis of ultrasonic sensors and prevents the wheelchair from colliding with objects in the

environment. No additional assistive functionality will be given to the user. Due to the ongoing human-system interaction and communication, the technical system is capable of recognizing changes in the current ability level of its user, for example, due to the confrontation with a stressful situation. If this is the case, the system changes its mode and activates an autonomous navigation mode, which does not only prevent the wheelchair from colliding with moving and stable, positive and negative obstacles, but also drives the user autonomously to a – from him/her – desired goal position. In order to enter such a desired goal position, a touchscreen is mounted on the wheelchair, which offers different content representations. While, it could display a floor map of the apartment and request from the user to click on the position, he/she would like to be driven to; it could also in a first step display a list of rooms available in the apartment and if one room has been selected, a list of objects as goal positions could pop up, from which the correct one needs to be chosen by the user. Depending on the automatic assessment of the user's current abilities (cf. [12, 13]), which also underlies the activation of the assistive functionality mode, the system could define the content representation which can without great cognitive effort be interpreted by the user, such that the possibility of a wrong entry is reduced.

Hence, such and more detailed descriptions of how the system will be used from a broad range of users allows the engineers to reduce misunderstandings of the system requirements and offers a deep understanding of the system to be developed, being, thus, an important basis for the following system design step.

### 2.4 System Design

In order to actually realize the system as envisioned, a system design approach needs to be worked out. In order to support this step, it is recommended to use the component-based design process *KobrA* [14] and to enhance the process with methods for system architecture design [15, 16] and dependability assurance methods. This developed design process provides methods to define functional and non-functional properties, top-down design and bottom-up integration of features as well as methods for testing and assessing the system during run time (online monitoring). Because human-technology-interaction is more and more one of the most critical factors for designing dependable systems with human involvement, a special focus has been placed on specifying the interfaces between humans and technical systems. As statistics (cf. [17, 18]) demonstrate, in 1960 only about 20% of system failures could be attributed to the so-called human factor, this percentage has risen up to 90% in the 1990s.

The component based design method KobrA2.0 has been utilized during the wheelchair development process. The design method is based on orthogonal views of the system and components and on a strict separation of specification and realization. KobraA2.0 promotes stepwise component decomposition at different abstraction levels, components view levels, and components decomposition levels. It includes both "top-down" elements and "bottom-up" approaches, which are suitable for an efficient prototypal system realization. The generic design method is compatible with the developed system architectural concepts as well as with all relevant component

types. The possibility to define a quality level and built-in tests during the design process is an essential part of the seamless design method.

The Recursive Nested Behaviour-based Control (RNBC) Structure [15], possesses properties necessary for building a complex yet dependable system. The fixed structure and the hierarchical nesting of the behavioural levels (the lower, less complex behaviours are embedded within the higher, more sophisticated behaviours) ensure the stability and predictability of the system's behaviour. Due to the fact that interactions only take place between neighbouring levels (recursiveness), the communication effort is moderate and well-defined interfaces ease the implementation of different levels by co-operating work groups. Because of the recursive extensibility, prototypes built bottom-up are operational through-out all development stages.

The development process starts with the identification of the fundamental behaviours, i.e axis-level control, robot-level control, collision avoidance, local navigation, and global navigation. The behaviours are sorted according to the required dynamics starting from the slowest behaviour on the top of the structure.

In the next step, the behaviours will be connected according to the required input and output signals within one level building one unique interface to neighboured levels. The behavioural levels will be connected recursively corresponding to Fig. 1 building the overall system structure of the wheelchair. Additional to the functional interfaces the behavioural levels provide interfaces for system monitoring and reconfiguration.

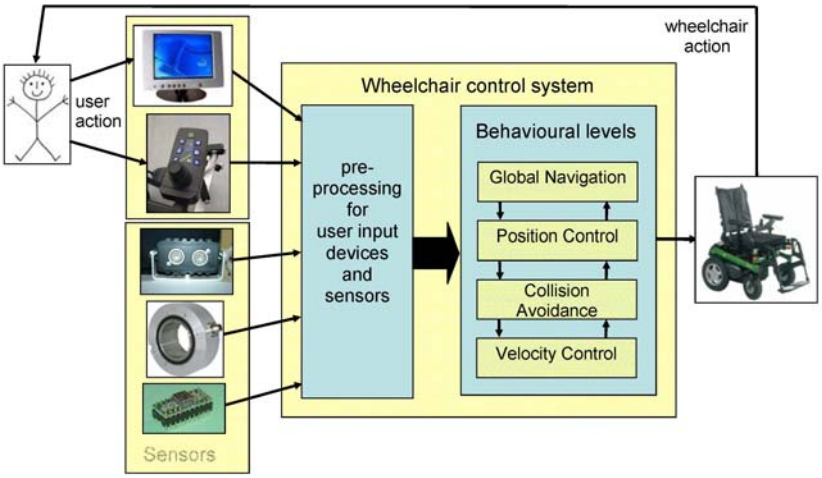



**Fig. 1:** Control system of the assisted electrically powered wheelchair

This overall system structure, can be utilized further on in the KobrA2.0 component specification process, while the behavioural levels are related to the system decomposition phases (in each phase one new level is tackled) and the behaviours within one level are related to the component decomposition (each behaviour states one basic component, which may be separated into functional components at the bottom of the decomposition process).

### 2.5 System Implementation

Since the assistance wheelchair is based on a commercial electrically powered wheelchair (OttoBock Healthcare GmbH), the mechanical setup, and some further components and behaviours are predefined, e.g. the axis-level velocity control behaviour. This must be considered in the definition of interfaces, the realization of upper level behaviours and the integration of components.

The overall system structure must also be reflected by the sensor configuration on the corresponding behavioural level. The velocity measurement is enhanced by incremental encoders on the wheel axes and by a gyro measuring the angular rate of the wheelchair orientation. The ultrasonic sensors are arranged around the wheelchair in order to detect a broad class of possible obstacles. However, for geometrical and physical reasons not all kinds of obstacles can be detected by ultrasonic sensors, e.g. holes in the floor or stairs. In order to avoid critical situation during backward driving additional infrared sensors are mounted on the rear side of the wheelchair, which are able to detect descending stairs.

According to the system architecture (Fig. 1) the behavioural levels and the corresponding components can be realized separately, which is described in the following.

*Axis-level velocity control*

The axis-level velocity control is a pre-fabricated component, which is integrated in a separate control system. It consists of a cascaded control structure for motor current control, velocity estimator and a feedback velocity control for the single driven wheels. In the basic system the input signal originates from the joystick output. The joystick provides the reference velocity vector (magnitude, angular rate), which is transformed into axis-level references using the inverse kinematics of the wheelchair. Depending on the selected mode, the joystick signal is modified by upper level behaviours.

*Robot-level velocity control*

The robot-level velocity control uses the reference velocity from the upper level and the velocity sensor signals (encoder and gyro) to calculate the velocity error. This error is compensated by a proportional integrating (PI) controller. Since the velocity measurement is error sensitive against bias drift, slippage and mechanical errors both sensor values are fused, in order to combine the advantages of both sensors.

*Collision Avoidance / local navigation*

A reflexive collision avoidance behaviour is realized based on the artificial potential field method [19]. This method enables a fast reaction on moving obstacles without knowing the exact position of the objects. The original algorithm which determines concentric virtual forces $F_i$ has been enhanced by a momentum vector $M_{rot}$, which reflects the asymmetry of the wheelchair in relation to the centre of rotation (see Fig. 2.). According to the resulting forces and momentum the velocity reference coming from the upper level is modified and forwarded to the velocity control level in order to ensure a safe navigation.

*Local navigation*

The local navigation behaviour ensures, that the wheelchair is able to reach way points or goal positions using a fuzzy control structure. The reference positions are provided by the path planning behaviour. Since no global positioning is available for indoor navigation the position control uses the fused sensor signals from the wheel encoders, the odometry and a dead reckoning algorithm in order to calculate the actual position. The actual position may be updated if absoluteposition is provided by an additional sensor.

*Global navigation*

The global navigation behaviour is based on the A* path planning algorithm, which calculates the shortest way between a staring point and a goal point for a given topological-metric map of the environment.

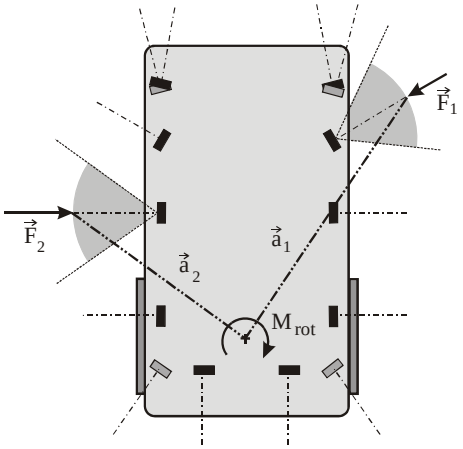



**Fig. 2:** Ultrasonic sensor configuration (small black and grey boxes), virtual forces and momentum calculated by the collision avoidance algorithm.

*User command interface*

The user command interface consists of a touchscreen and a conventional wheelchair joystick, which is adapted from the original wheelchair, i.e. the wheelchair driver can use the wheelchair in the non-assisted mode, without any drawbacks. Switching on the assisting system, the user is currently requested to input the mode, in which the wheelchair will operate. In the assisted mode the user is supported by the collision avoidance behaviour and the lower levels. In the full autonomous mode the user selects the goal position from a set of pre-defined goals using the touch panel. In a planned extension the automatic recognition of user capabilities and selection of the suited mode will be implemented into the user command interface.

## 2.6 System Integration and Test

In order to integrate all behaviours described above in a dependable way, a suitable hardware and software system has been setup. Due to the behavioural levels a separation of functionality and a distribution over many components is possible. For the specific system an industrial control PC running the realtime operating system QNX has been selected. The PC is equipped with interface cards for CAN, Ethernet, WLAN, and I²C communication as well as with arbitrary digital and analog channels. The behaviours are integrated as software components which are executed in form of separated processes within the realtime system (Fig. 3. shows lowest three levels).

The behavioural components communicate with each other using interface threads. This ensures the realtime communication without data blocking or collision. The sensor hardware is connected over special drivers. While the behaviours are encapsulated the interfaces are freely accessible form outside. This can be used for a local online-monitoring and reconfiguration process, which is implemented in the next higher behaviour level. The advantages arising from the separation of behaviours has also been used during the functional test of software components.

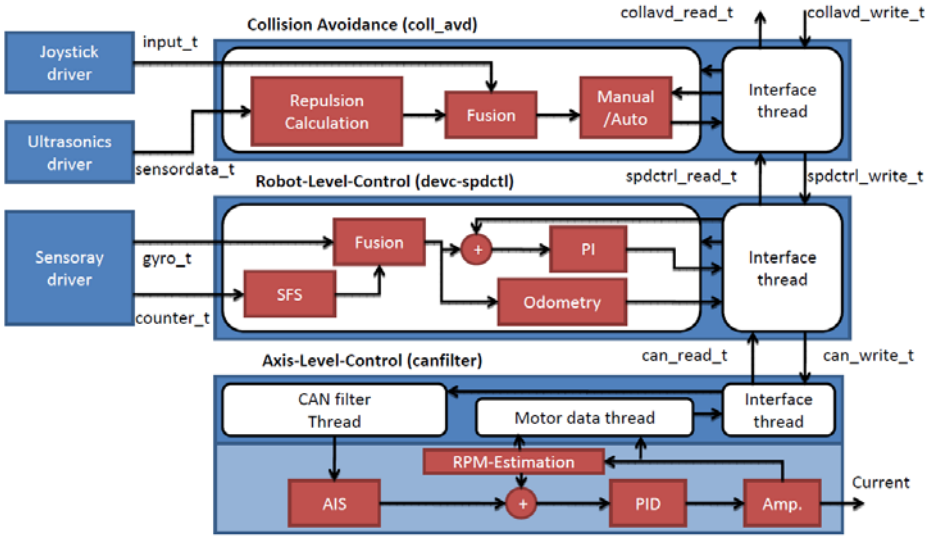



**Fig. 3:** Software implementation of the wheelchair assistance system.

Thus, the implementation maintains all aspects of the generic system architecture:

- the implementation is flexible due to the free choice of methods and components for the implementation of single behaviours
- the structure is extensible enabling the adding or removal of behaviours
- the signals from and to the layers can be observed locally in order to reduce the development effort due to sparse modelling and communication effort in the running system
- the behavioural levels can be developed and tested separately, ensuring high maintainability

- the user interface is distributed (touch panel for higher levels, joystick on the velocity level) ensuring the input of appropriate signals on the corresponding behavioural level

### 2.7 System Evaluation

In a last step, the resulting system needs to be evaluated. More specifically, it is to be tested whether the needs, which the system should at least partially reduce, has been met with the system at hand. A systematic approach for an evaluation is provided e.g. by [2] and uses a variety of research methodologies (cf. [6]). In parallel to the procedure described for the user needs assessment is it desirable to combine different methods to yield a greater validity of the results.

With regard to a quantitative evaluation, an experimental set-up can be taken, during which the participants are grouped on the basis of random numbers to avoid systematic selection effects. While a control group executes standardized tasks with a standard wheelchair, an experimental group should perform the same tasks with the new assistance system for powered wheelchair control. During this experiment, a set of variables of interest should be measured, which reflect appropriate operationalizations of the needs.

With regard to a qualitative evaluation, the user's opinions on the new system in comparison to the standard off-the-shelf system can be assessed for example with appropriate available questionnaires or with especially for these purposes constructed questionnaires (cf. [4]).

Due to the great sample size, which is required to yield a high power of the results for these types of evaluations (cf. [20]), we did not use such a between-subject manipulation but a within-subject evaluation. This means, each participant was tested twice – once with the standard system and once with the new assistance system for powered wheelchair control. Such an evaluation procedure has the advantage that the variance, which can be contributed to the subject itself, can be controlled by applying a repeated measurement statistical analysis (cf. [21]).

Such a procedure has been conducted with regard to the wheelchair application and more specifically for evaluating the autonomous navigation behavior. For this purpose, about 20 participants drove through a standardized course twice, once with a two-switch control, once with an autonomous navigation behavior activated. While measuring quantifiable data such as the distances driven and the times required reaching a specific position, a usability questionnaire has been applied in addition in order to gather data on how the participants liked an autonomously driving assistance system. Especially the data on the usability questionnaire demonstrate the superiority of the autonomous navigation mode: nearly in all aspects (i.e. in easiness to learn, intuitiveness, safety, and comfort) the autonomous navigation mode outperformed the manual driving mode.

While this reflects an evaluation of one part of the system, i.e., the autonomous navigation mode, a study evaluating the overall system, which adapts its functionality to the user's abilities, will be conducted in the near future. Such an evaluation will then also give important feedback on this dependability-centered system design approach.

## 3 Conclusion

This paper aimed at introducing a design approach for dependable complex computing systems considering hardware, software, and human factors. For this purpose, research design methods from the psychological field of formative and summative evaluation (required for the user needs assessment and the final evaluations) has been combined with software tools (e.g., development of use cases, modeling of software components) and implemented in a traditional system design approach covering the development of a system architecture, implementation, integration, and test. On this basis a set of design steps have been introduced which start with a user needs assessment, during which qualitative and quantitative methods are applied in order to identify a need, which the future system should reduce and a system requirement analysis, which defines the functional and non-functional properties of the computing system. On that basis, use cases were developed, which reflect the prototypical usage of the system at hand and which aim at clarifying the chosen design solution. With such a clear vision in mind, a system architecture and control structure can be developed, which is the starting point for the system development. After having implemented and integrated the system, a thorough test phase will ensure that the system meets its specifications. If this phase can be completed successfully, the proposed system development process finishes with a summative evaluation analyzing whether the system is actually capable of reducing the - in the user needs assessment - identified needs. In order to clarify these different steps, an example of an assistance system for electrically powered wheelchairs has been chosen and the results of each of these steps has been summarized in this paper – demonstrating the potentials the proposed dependability-centered system design approach has especially on reducing the possibility of a failure of the human-automation system.

Future work will aim at completing the implementation of the overall system and at evaluating its final version as described in Section 2.7. During this final evaluation a special emphasis will be put on deriving benchmarks, which will enable a fair comparison with other system design approaches especially with regard to dependability. For this purpose, the number of accidents which occurred when using the system in the long run could be compared with the number of accidents when using a system which development was based on another design approach. Other factors which are also indicators about the success of the dependability-centered system design approach can be considered as well. Hence, the evaluation will not only enable judging on the resulting wheelchair system, but also at rating the proposed system design approach and at demonstrating the potential benefits of a design approach considering all aspects of a complex human-technology system consisting of not only hardware, software or human factors, but the interaction of these system components.

## References

1. Kephart, J. O., Chess, M.: The vision of autonomic computing. IEEE Computer, pp. 41-50, 2003.
2. Rossi, P.H., Freeman, H.E., Lipsey, M.W.: Evaluation: A systematic approach. London: Sage Publication, 1999.
3. Scholz, R. W., Tietje, O.: Embedded case study methods: Integrating quantitative and qualitative knowledge. Thousand Oaks: Sage Publications, 2002.
4. Trochim, W.: The research methods knowledge base, 2nd Edition. Atomic Dog Publishing, Cincinnati, OH, 2000.
5. Campbell, D. T. , Fiske, D. W.: Convergent and discriminant validation by the multitrait multimethod matrix. Psychological Bulletin, vol. 56, no. 2, pp. 81-105, 1959.
6. Shadish, W.R., Cook, T.D., Campbell, D.T.: Experimental and quasi-experimental designs for generalized causal inference. Boston: Houghton-Mifflin, 2002.
7. Jipp, M., Bartolein, C., Badreddin, E.: Assisted wheelchair control: Theoretical advancements, empirical results, and technical implementation. Proceedings of the International Symposium on Mechatronics and its Applications, 4, ISMA01-ISMA07, 2007.
8. Jipp, M., Bartolein, C., Badreddin, E.: Quantitative comparison of the joystick control mode and the two-switch control mode when steering a wheelchair. Accepted for Publication at the Annual Meeting of the Human Factors and Ergonomics Society, 2009.
9. Jipp, M., Wittmann, W. W., Badreddin, E.: Concurrent validity of individual differences in intelligence in activity differences of handicapped wheelchair users. Proceedings of the Annual Meeting of the Human Factors and Ergonomics Society, 52, 990-994, 2008.
10. Jipp, M., Bartolein, C., Badreddin, E., Abkai, C., Hesser, J.: Psychomotor profiling with Bayesian Networks: Prediction of user abilities based on inputs of motorized wheelchair parameters. Accepted for Publication of IEEE International Conference on Systems, Man, and Cybernetics, 2009.
11. Bittner, K., Spence, I.: Use case modeling. Addison-Wesley Pearson Education, Boston, 2003.
12. Wagner, A., Bartolein, C., Jipp, M., & Badreddin, E. (2008). Assessment of the user's dependability-relevant abilities for enhanced human-technology interaction. Proceedings of the Annual Meeting of the Human Factors and Ergonomics Society, 52, 980-984.
13. Jipp, M., Badreddin, E., Abkai, C., & Hesser, J. (2008). Individual ability-based system configuration – Cognitive profiling with bayesian networks. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1, 3359-3364.
14. Atkinson, C., Brenner, D., Falcone, G., Juhasz, M.: Specifying high-assurance services. IEEE Computer, vol. 41, no. 8, pp. 64-71, 2008.
15. Badreddin, E.: Recursive control structure for mobile robots. International Conf. on Intelligent Autonomous Systems 2 (IAS.2), Amsterdam, 11-14, 1989.
16. Bartolein C., Wagner, A., Jipp, M., Badreddin, E.: Multilevel intention estimation for wheelchair control, Proc. of the European Control Conference 2007, Kos, Greece, July 2-5, 2007.
17. Hollnagel, E.: Human reliability analysis: Context and control. London: Academic Press, 1993.
18. Swain, A. D.: Human reliability analysis: Need, status, trends, and limitations. Journal of Reliability Engineering and System Safety, 29, 301-313, 1990.
19. Badreddin, E.: Control and System Design of Wheeled Mobile Robots, Habilitationsschrift, ETH Zürich, 1995
20. Cohen, J.: Statistical power analysis for the behavioral sciences, 1988.
21. Bortz, J.: Statistik für Human- und Sozialwissenschaftler. Springer. Berlin, 2005.

# Dependable component-based design on the Example of a Heating Control System

Leila Zouaghi[1], Markus Koslowski[1], Alexander Alexopoulos[1], Florian Barth[2], Meike Jipp[1], Raul Fajardo[3], Yi Luo[1], Achim Wagner[1], Essam Badreddin[1]

[1]Automation Laboratory, University of Heidelberg, Germany
{leila.zouaghi, markus.koslowski, alexander.alexopoulos, meike.jipp, yi.luo, achim.wagner, badreddin}@ziti.uni-heidelberg.de
[2] Chair of Software Engineering, University of Mannheim, Germany
barth@informatik.uni-mannheim.de
[3] Department for Application Specific Computing, University of Heidelberg, Germany
raul.fajardo@ziti.uni-heidelberg.de

This poster illustrates new methodologies for the design and the realization of dependable component-based systems covering hardware, software and human factor aspects. The system structure uses the approach of the "Recursive Nested Behavior Control" (RNBC) ensuring dependable operation and seamless interaction of the system's components [1]. For the design and the specification of the system the component based design Method KobrA [2] is applied. The specification of hardware components based on high-level hardware design, Transaction-Level Model [3], is presented. Dependability relevant concepts such as Quality of Service and built-in tests using test- sheets [4] as a new way of defining the expected functionality of a component are introduced. For the on-line monitoring of the system we propose a particle Petri net model. This model allows the estimation of the hybrid state of the system and the detection of discrepancies between the expected nominal behavior of the system and the observed one. For a better reflexion of the reality we model both discrete and continuous state of the behaviour. An integrated dependability model has been developed which includes system, hardware, software, and human properties on a behavioural view. It defines, how much the system's behaviour deviates from the desired behaviour over the system's mission (usage) and how much the system's behaviour keeps away from the non-desired (critical) behaviour. A literature review has shown that quantitative descriptions of system dependability are generally done over combinations of some attributes [5], [6]. These attributes are: reliability, availability, safety, integrity, confidentiality and maintainability. We introduce a behaviour based modelling approach [7], [8], [9]. A dependability metric was developed, which can be used during design respectively during run-time to measure the sub-systems' as well as the overall system's dependability. To be able to measure the involved dependability attributes during run-time built-in test software modules have been generated based on test-sheets.

Approaches for the design of Human-Technology Interaction adapt the technical system to the operator only in a very general way and ignore differences between operators. We adapt the technical system and its interface to the abilities of the operator and take into account the individual differences in these abilities between and within operators [10]. The interfaces are adapted so that its demand character

does not exceed the ability level of the operator. While for some operators it is easier to interpret figural information, figural information will be displayed. Others prefer verbal and numerical information, so that for them, the relevant contents are displayed in the numerical and verbal representation.

In order to demonstrate the feasibility of the proposed methods and techniques (system architecture, dependability modeling and measure, component-based software and hardware design, testing using test sheets, monitoring, human-technology interaction etc.) we use a simple case study from the control engineering domain: Heating Control System (HCS), which is responsible for maintaining a comfortable temperature in a house by regulating the temperature of the available radiators. Using this case study, in which scientist from different disciplines and application domains were involved, the developed methods were tested and adapted. The achieved interesting results show the feasibility of the proposed methods and their usability for the design and the realization of dependable systems.

## References

[1] Badreddin, E. "Recursive Control Structure for Mobile Robots", International Conf. on Intelligent Autonomous Systems 2 (IAS.2), Amsterdam, pp. 11-14, 1989.

[2] Atkinson C., Bostan P., Brenner D., Falcone G., Gutheil M., Hummel O., Juhasz M. & Stoll D. (2008). Modeling Components and Component-Based Systems in KobrA , to appear in A. Rausch, R. Reussner, R. Mirandola, F. Plasil (eds.): The Common Component Modeling Example: Comparing Software Component Models, Springer

[3] Cai, L., Gajski, D., Transaction level modeling in system level design, Tech. Rep., Center for Embedded Computer Systems, Irvine, Calif, USA, 2003.

[4] Atkinson C. & Brenner D. (2008). Software Testing using Test Sheets, submitted to International Symposium on Software Testing and Analysis (ISSTA), Seattle, Washington, July 20-24 2008

[5]Laprie, J. C. Dependable computing: BASIC concepts and terminology: in english, french, german, italian and japanese. Ed. Springer – Verlag, 1992

[6]Laprie, J.C., (1995). Dependable Computing and Fault Tolerance: Concepts and Terminology. Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years'., Twenty-Fifth International Symposium on, p. 2.

[7]Rüdiger, J., Wagner, A., & Badreddin, E. (2007a). Behavior based definition of dependability for autonomous mobile systems. European Control Conference (July, 2007). Kos, Greece.

[8]Rüdiger, J., Wagner, A., & Badreddin, E. (2008a). Behavior based dependability estimation. ICINCO. Funchal, Madeira - Portugal.

[9]Rüdiger, J., Wagner, A., & Badreddin, E. (2008b). Behavior based estimation of dependability for autonomous mobile systems using particle filter. IFAC. Seoul, Korea.

[10] Jipp, M., Wagner, A., & Badreddin, E. (2008), Individual Ability-Based System Design of Dependable Human-Technology Interaction. International Proceedings of the IFAC World Congress in Seoul, South Corea.