Self-Optimizing Mechatronic Systems and Safety Standards: Challenges and Limits

5. Bieleschweig Workshop

Robert Traussnig
Dr. Holger Giese

Munich, April 5-6, 2005
Agenda

I. Motivation
II. Self-Optimizing Mechatronic Systems
III. Safety Challenges
IV. Research Concepts
V. Limitations of Standards
VI. Industry Approach
VII. Summary
I. Motivation

- increasing complexity in safety-critical systems
- replacement of hardware by software
- increased demand for quality on software
- markets demand faster innovation cycles

introduction of new technologies:
- model-based software development
- automated code-generators
- self-optimization
II. Self-Optimizing Mechatronic Systems

**Mechatronic systems**

- mechanics
- electronics
- control engineering
- software

**Self-Optimization**

- systems endogenously modify their objectives in response to changing conditions
- adapt their parameters, structure and behavior to fulfill their objectives
II. The railcab Project – SFB 614

Collaborative Research (SFB 614) at Uni Paderborn:
Create a system of collaborating, self-optimizing, autonomous track-based, high-speed shuttles for passenger and cargo transportation.
II. The railcab Project

Movie
III. Safety Challenges

Challenge: System Complexity

• Multiple layers
  – fleet management
  – convoys, shuttles, section control
  – active suspension/tilt module
  – actors/sensors

• “Self-Optimization” at each layer (context dependent) by the loop:
  (1) sense environment $\Rightarrow$ (2) adjust goals $\Rightarrow$ (3) adapt behavior
Challenge: Multiple Disciplines

System behavior:
- Software-Agents for Logistics
- Real-time coordination
- Energy management
- Motion control

Goal: Design approach for the self-optimizing software of technical systems (which ensures safe coordination and online-reconfiguration)
IV. Research Concepts

Model-Based Development

Modeling → Model → Analysis → Synthesis → Model

Code
IV. Research Concepts

Operator Controller Module (OCM) Architecture
IV. Research Concepts

Online Reconfiguration

Layer: shuttles + section control

![Diagram showing reference trajectory and comfort measurement](image)
Online Reconfiguration

Layer: Suspension/Tilt Module

Online-reconfiguration via modes:

- Reference (use given trajectory), Absolute (use body acceleration), and Robust (requires only standard inputs) ⇒ different inputs required
IV. Research Concepts

Role-based Approach

Distance Coordination

:Shuttle

S2 -> S1
V. Limitations of Standards – IEC 61508

IEC 61508

- railcab certification wrt software based on IEC 61508?
- is it possible?
- are there any limitations?
V. Limitations of Standards – IEC 61508

IEC 61508-3, Annex A (normative)

Table A.2 Software design and development: software architecture design:

Technique #5: Artificial Intelligence / Fault correction for SIL 2,3,4: Not Recommended

Technique #6 Dynamic Reconfiguration for SIL 2,3,4: Not Recommended

What if the basis of the system is AI and Dynamic Reconfiguration?
V. Limitations of Standards – IEC 61508

IEC 61508-3

7.9.2.12 „Code verification: the source code shall be verified by static methods to ensure conformance to the specified design of the software module, the required coding standards, and the requirements of safety planning“
Note: In the early phases of the software lifecycle, verification is static (for example inspection, review, formal proof etc.)

Qualified Code Generator (QCG): generated code is correct-by-construction
No verification necessary!
V. Limitations of Standards – IEC 61508

IEC 61508-7

IEC 61508-7, C.4 Development tools and programming languages

C.4.3 Certified tools and certified Translators:

Whenever possible, tools should be certified...

To date, only compilers (translators) are regularly subject to certification procedures; these are laid down by national certification bodies and they exercise compilers (translators) against international standards such as those for Ada and Pascal.

Who will certify the MBD Environment/AGC and against what criteria?
V. Limitations of Standards – IEC 61508

IEC 61508-3

IEC 61508-7, C.4 Development tools and programming languages

C.4.4 Tools and translators: increased confidence from use

A translator is used, where there has been no evidence of improper performance over many prior projects.

When is „increased confidence“ good enough?
Cross-standard certification possible?
V. Limitations of Standards – DO-178B

RTCA DO-178B

- Civil Aviation Standard, U.S. Federal Aviation Administration
- in Europe DO-12B standard
- introduced in 1992
Qualification Requirements of the Automated Code Generator (ACG) with respect to DO-178B:

ACG defined as:
„Tool whose output is part of the airborne software and thus can introduce errors“

DO-178B, section 12.2.1:
„If a software tool is to be qualified, the software development processes for the tool should satisfy the same objectives as the software development processes of airborne software.“

„The software level assigned to the tool should be the same as that for the airborne software it produces.“
V. Limitations of Standards – DO-178B

**Qualifiable:** Tool has been developed in such a way that it is “prequalified” or „qualifiable“ which means that it is ready for qualification on specific projects.

**Qualified:** On a per-project basis only. Tool Criticality Level has to match the final Software Criticality Level.

**Certified:** Legal recognition by the certification authority that a product, service, organization or person complies with the authorities requirements.
VI. Industry Approach: Airbus Industries

On-Board Software

Source: Esterel Technologies
VI. Industry Approach: Airbus Industries

**Automated Code Generation at Airbus**

- cost of a minor bug detected in flight is between $100K - $500K
- cost of a major bug detected in flight is between $1M - $500M

*Airbus decided in the early 80’s to introduce automated code generation (ACG)*
VI. Industry Approach: Airbus Industries

A340/600 FCSC (Flight Control Secondary Computer):

70% automatically generated code
50% reduction in software development cost
reduction in modification cycle time by factor 3
VI. Industry Approach: Airbus Industries

Errors detected per 100 KBytes of code

- A 310 (1970s)
- A 320 (1980s)
- A 340 (1990s)

70 % ACG Code

Source: [7] p. 6
“No software bug ever detected in flight (including flight test) since the beginning of the use of automated code generator for fly-by-wire software.”

[10] F. Pothon, Airbus France
VII. Summary

- Self-Optimization: enormous potential, but how can we prove it’s safe?

- New development methods for safety critical systems:
  - Model-Based Development
  - Automated Code Generation
  - Online Reconfiguration

- Complex / not applicable certification processes slow down introduction

- Need for new verification/validation approach

- Current standards, especially IEC 61508 need to be adapted
Thank you for your attention!


References cont´d


