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

- 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

Mechatronic systems

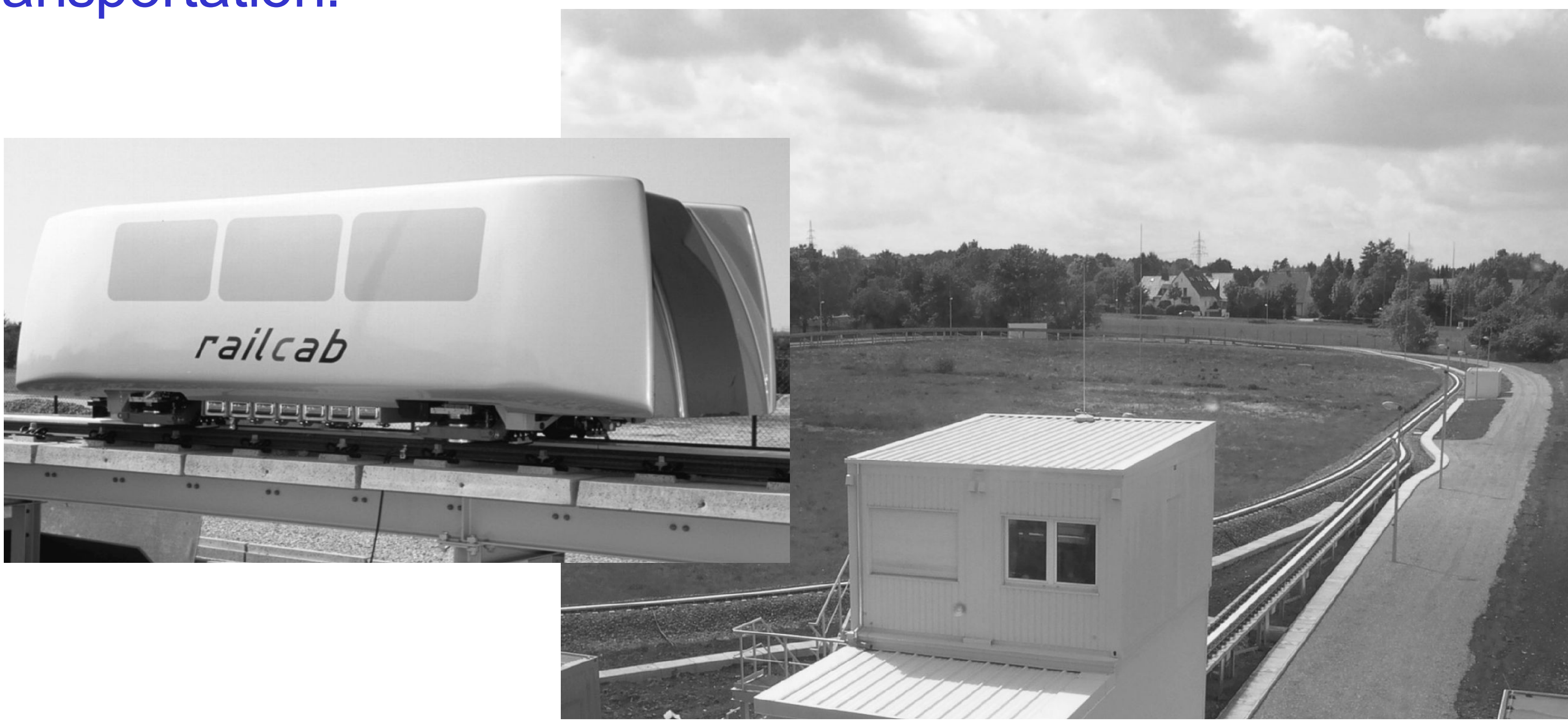
- mechanics
- electronics
- control engineering
- software

Self-Optimization

- systems endogenously modify their objectives in response to changing conditions
- adapt their parameters, structure and behavior to fullfill their objectives

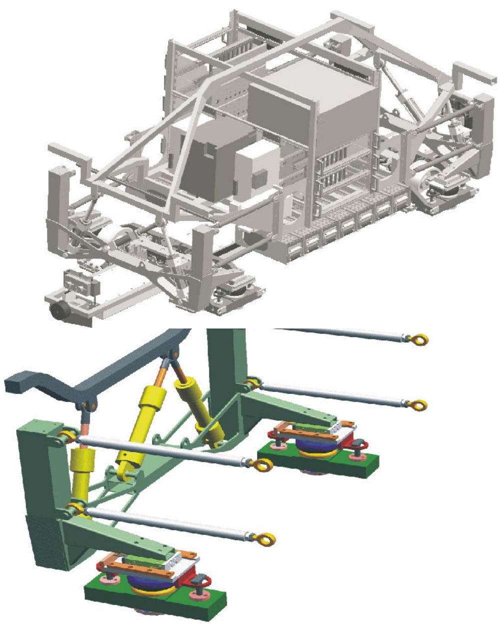
Collaborative Research (SFB 614) at Uni Paderborn:

Create a system of collaborating, self-optimizing, autonomous track-based, high-speed shuttles for passenger and cargo transportation.



Movie

Challenge: System Complexity



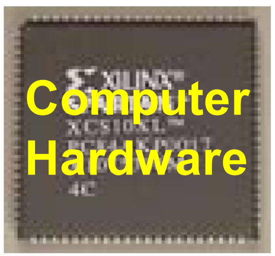
- Multiple layers
  - fleet management
  - convoys, shuttles, section control
  - active suspension/tilt module
  - actors/sensors

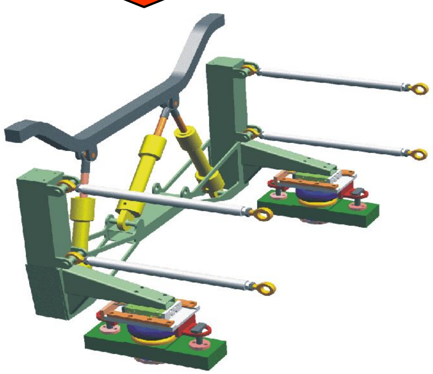
- “Self-Optimization” at each layer (context dependent) by the loop:

  (1) sense environment ⇨ (2) adjust goals ⇨ (3) adapt behavior

Challenge: Multiple Disciplines

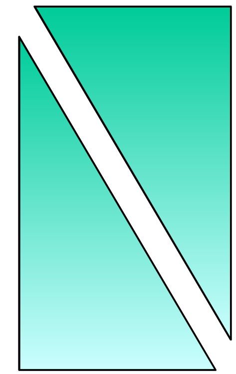
Software





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

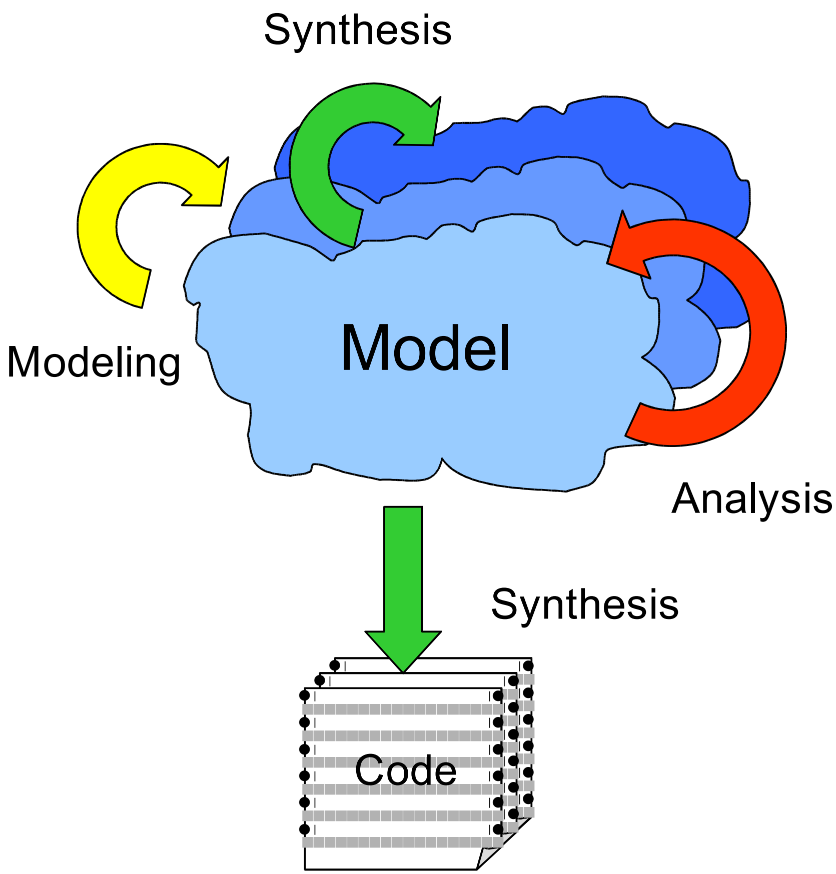
Software engineering



Control engineering

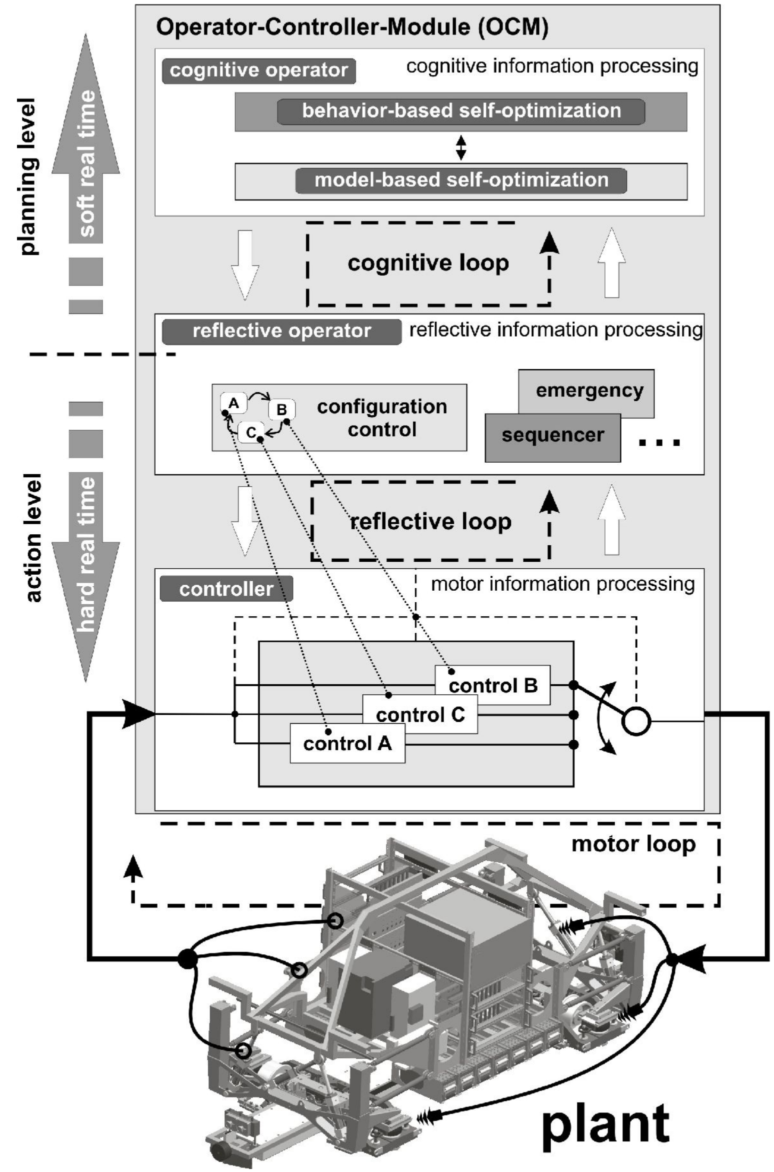
Goal: Design approach for the self-optimizing software of technical systems (which ensures safe coordination and online-reconfiguration)

Model-Based Development



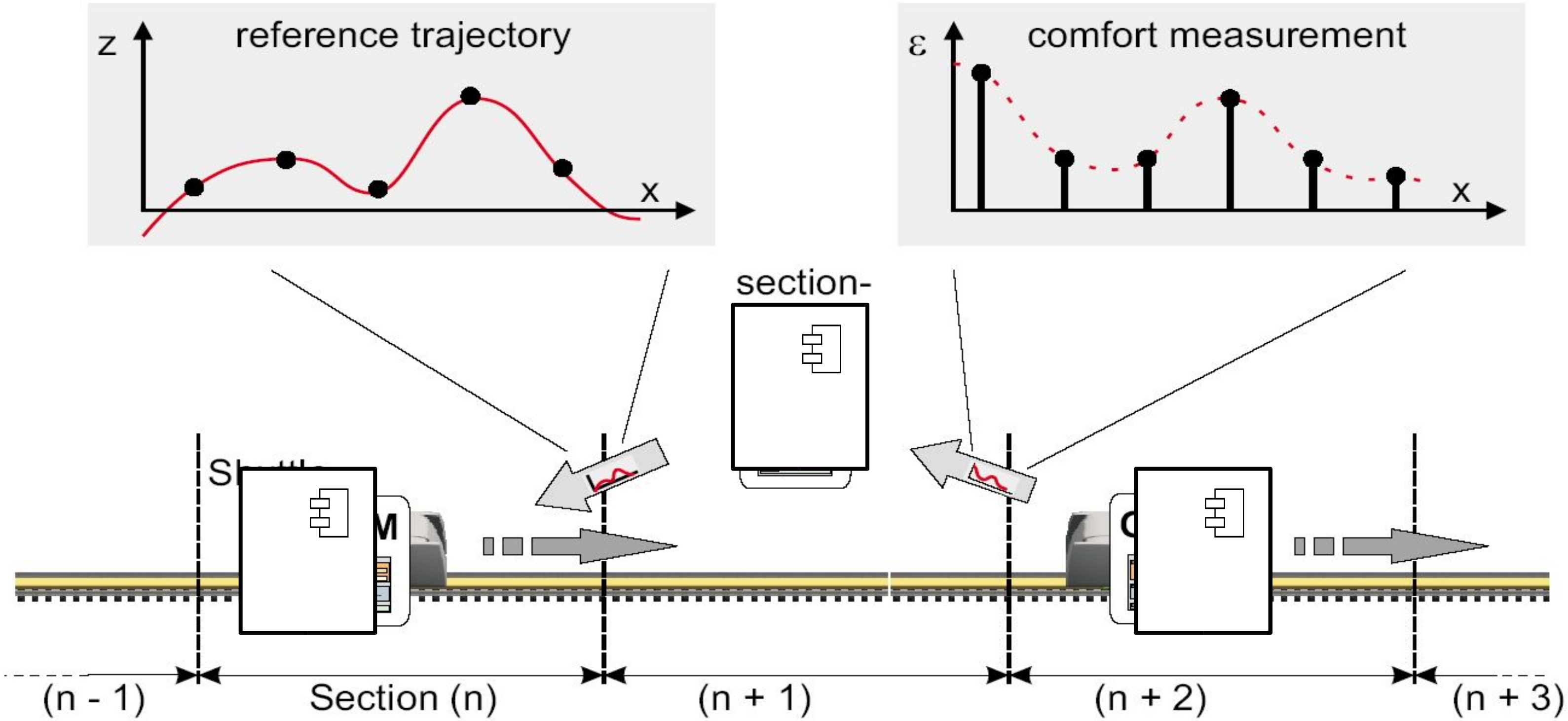
IV. Research Concepts

Operator Controller Module (OCM) Architecture



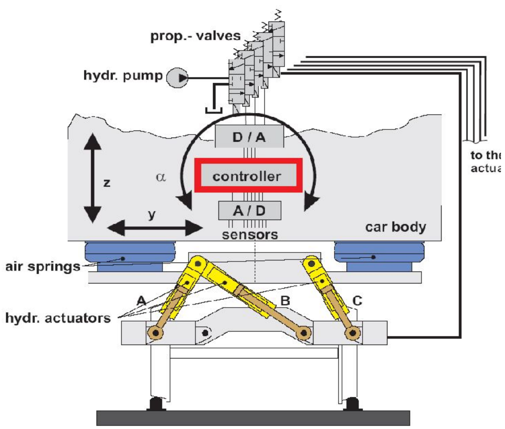
Online Reconfiguration

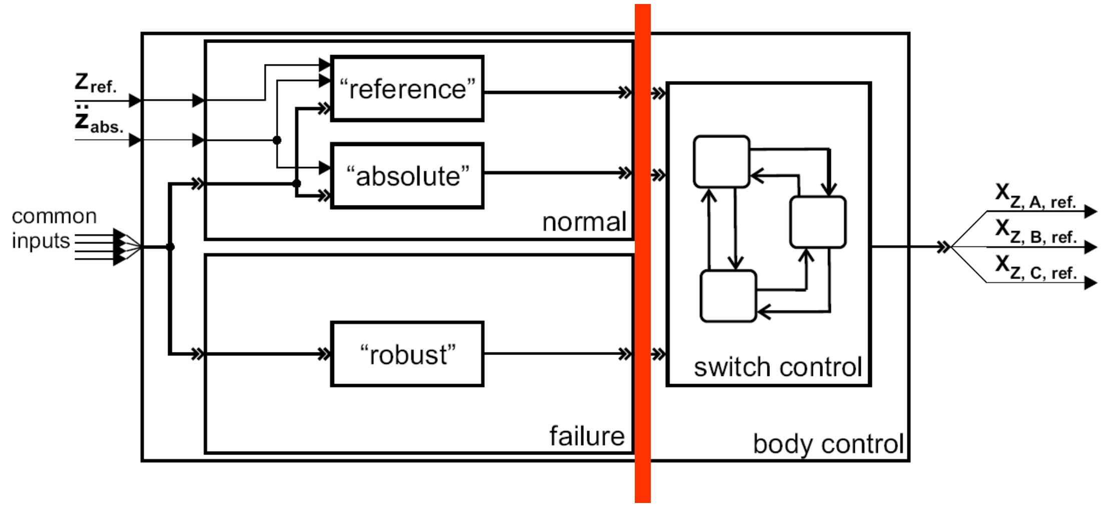
Layer: shuttles + section control



Online Reconfiguration

Layer: Suspension/Tilt Module

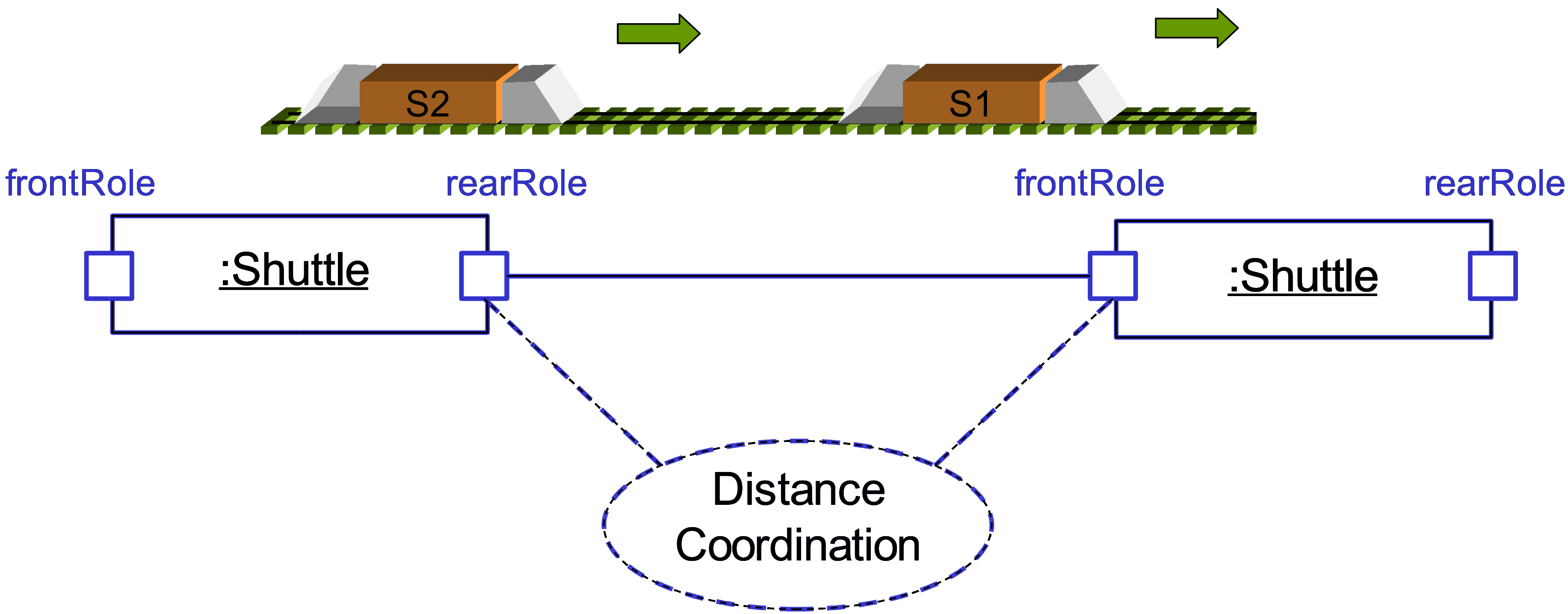




Online-reconfiguration via modes:

- Reference (use given trajectory), Absolute (use body acceleration), and Robust (requires only standard inputs) ⇨ different inputs required

Role-based Approach



IEC 61508

- railcab certification wrt software based on IEC 61508?
- is it possible?
- are there any limitations?

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?

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!

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?

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?

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.“

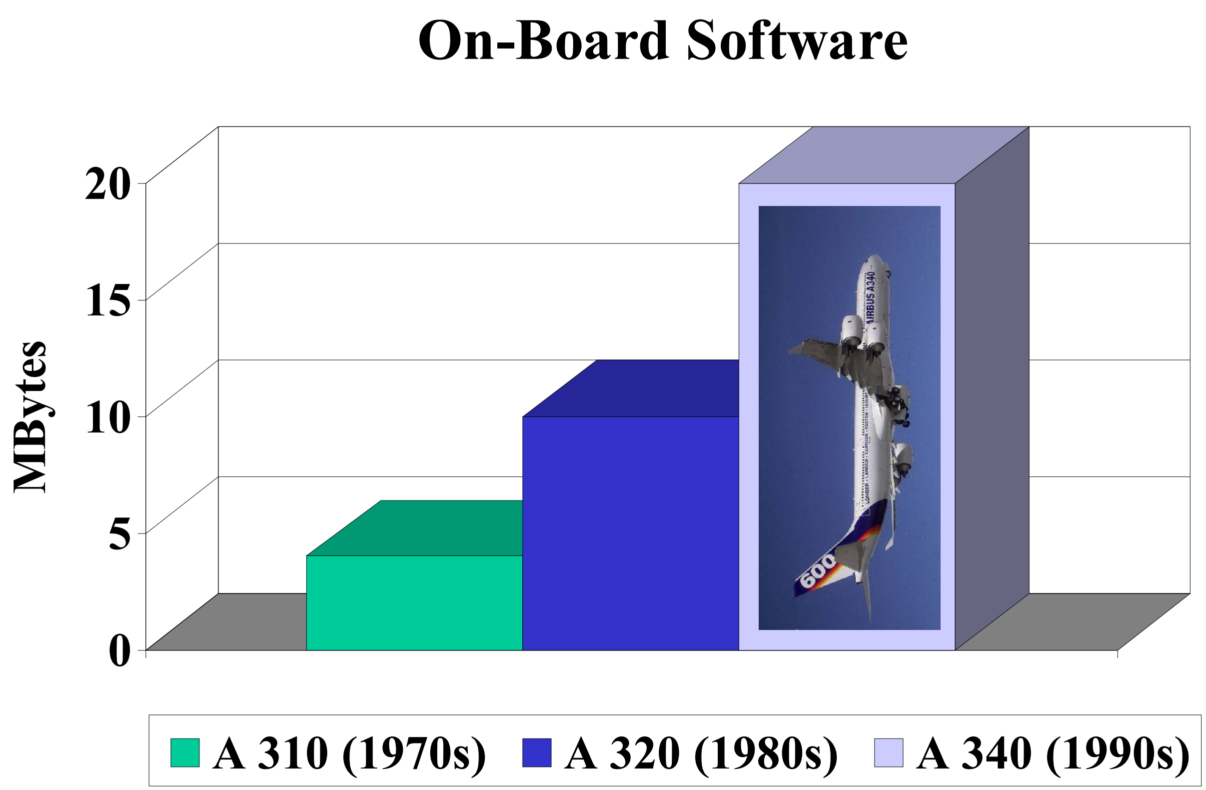
V. Limitations of Standards – DO-178B

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



Source: Esterel Technologies

University of Paderborn
Software Engineering Group

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

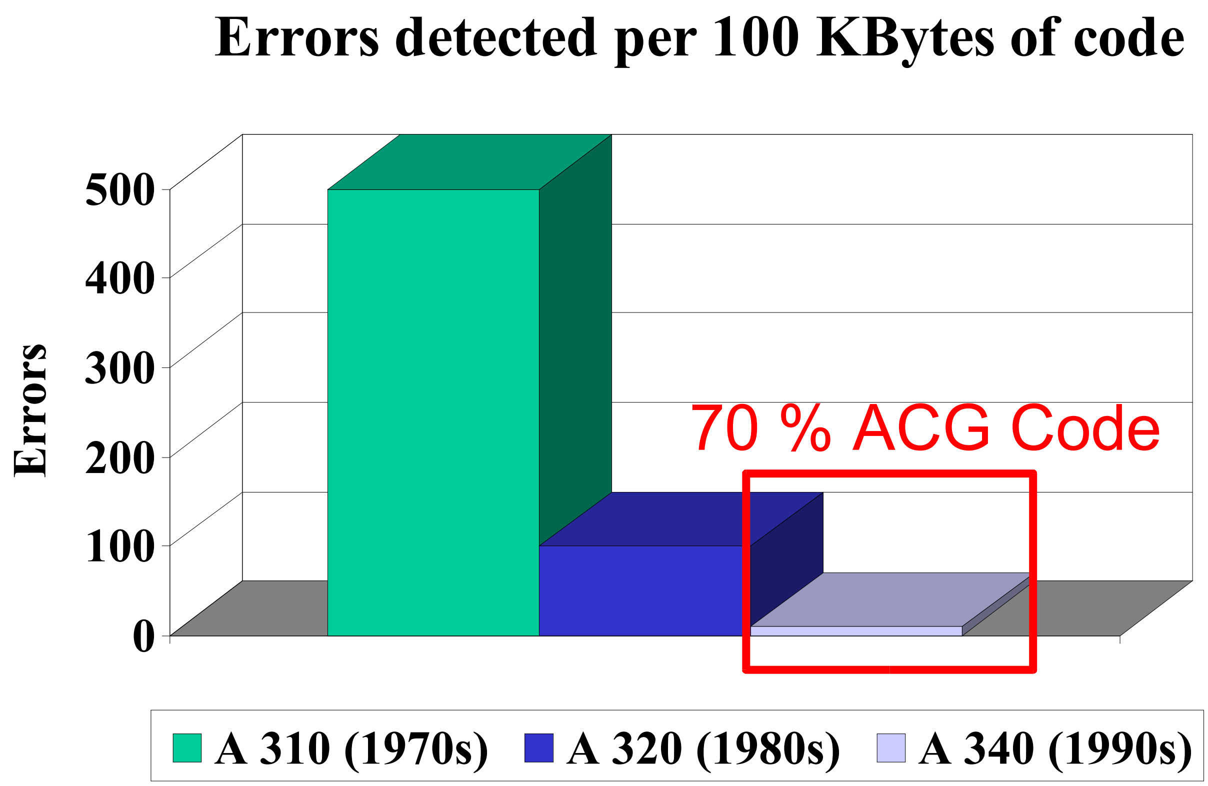
→ Airbus decided in the early 80‘s to introduce automated code generation (ACG)

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



Source: [7] p. 6

VI. Industry Approach: Airbus Industries

“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

[1] J. McDermid, D. Pumfrey: Software Safety: Why is there no Consensus? In: Proceedings of the 19th International System Safety Conference, Huntsville, AL, USA (2001) 17–25

[2] S. Burmester, H. Giese, and O. Oberschelp: Hybrid UML Components for the Design of Complex Self-optimizing Mechatronic Systems. In Informatics in Control, Automation and Robotics, Kluwer Academic Publishers, 2005. (to appear)

[3] H. Giese, S. Burmester, W. Schäfer, and O. Oberschelp: Modular Design and Verification of Component-Based Mechatronic Systems with Online-Reconfiguration. In Proc. of 12th ACM SIGSOFT Foundations of Software Engineering 2004 (FSE 2004), Newport Beach, USA, ACM, November 2004

[4] H. Giese, M. Tichy, and D. Schilling: Compositional Hazard Analysis of UML Components and Deployment Models. In Proc. of the 23rd International Conference on Computer Safety, Reliability and Security (SAFECOMP), Potsdam, Germany, Lecture Notes in Computer Science (LNCS), Springer Verlag, September 2004.

[5] H. Giese, M. Tichy, S. Burmester, W. Schäfer, and S. Flake: Towards the Compositional Verification of Real-Time UML Designs. In Proc. of the European Software Engineering Conference (ESEC), Helsinki, Finland, pp. 38--47, ACM Press, September 2003.

[6] S. Burmester, H. Giese, and W. Schäfer: Code Generation for Hard Real-time Systems from Real-time Statecharts. Tech. Rep. tr-ri-03-244, University of Paderborn, Germany, October 2003.

References cont´d

[7] J.-L. Camus and B. Dion: Efficient Development of Airborne Software. White Paper. Esterel Technologies, 2003. p6;33

[8] Debra S. Hermann: Software Safety and Reliability: Techniques, Approaches and Standards of Key Industrial Sectors. IEEE Computer Press, November 1999

[8] Nancy G. Leveson. *Safeware: Systems Safety and Computers*. Addison-Wesley, 1995.

[9] International Electrotechnical Commission: *International Standard IEC 61508*. First Edition. CEC/IEC 61508:1998

[10] F. Pothon: Automated Code Generation in Airbus. In Federal Aviation Administration / ERAU Software Tools Forum, Daytona Beach, May 2004.

[11] J. Gausemeier: From Mechatronics to Self-Optimization. In International Congress on CAD- FEM Technology, Friedrichshafen, October 2002.