



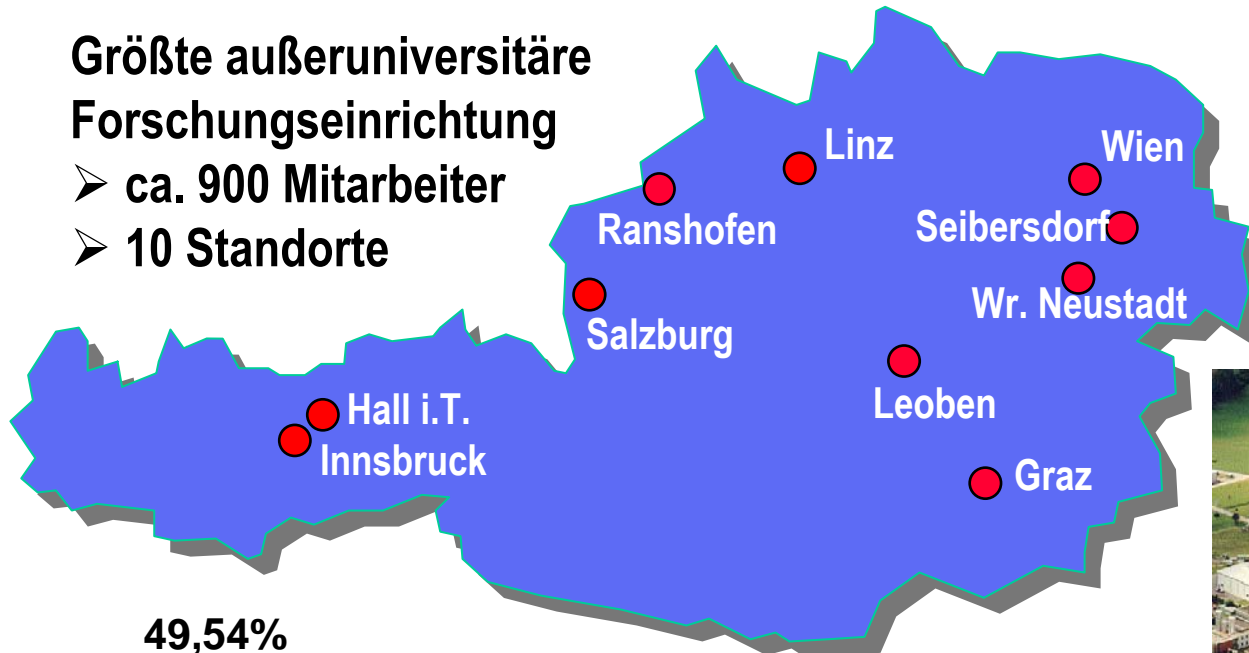
**Akkreditiertes Prüflabor – Wofür?  
Steigendes Sicherheitsbewusstsein im Bereich  
kritischer computergesteuerter Systeme**

**Erwin Schoitsch, Thomas Gruber**  
**Austrian Research Centers GmbH - ARC**  
**Smart Systems Division**  
{erwin.schoitsch, thomas.gruber}@arcs.ac.at

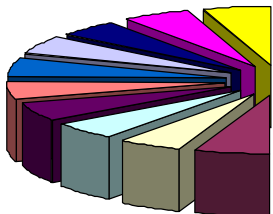
## Austrian Research Centers - ARC

Größte außeruniversitäre  
Forschungseinrichtung

- ca. 900 Mitarbeiter
- 10 Standorte



49,54%  
Wirtschaft



50,46%  
Bund

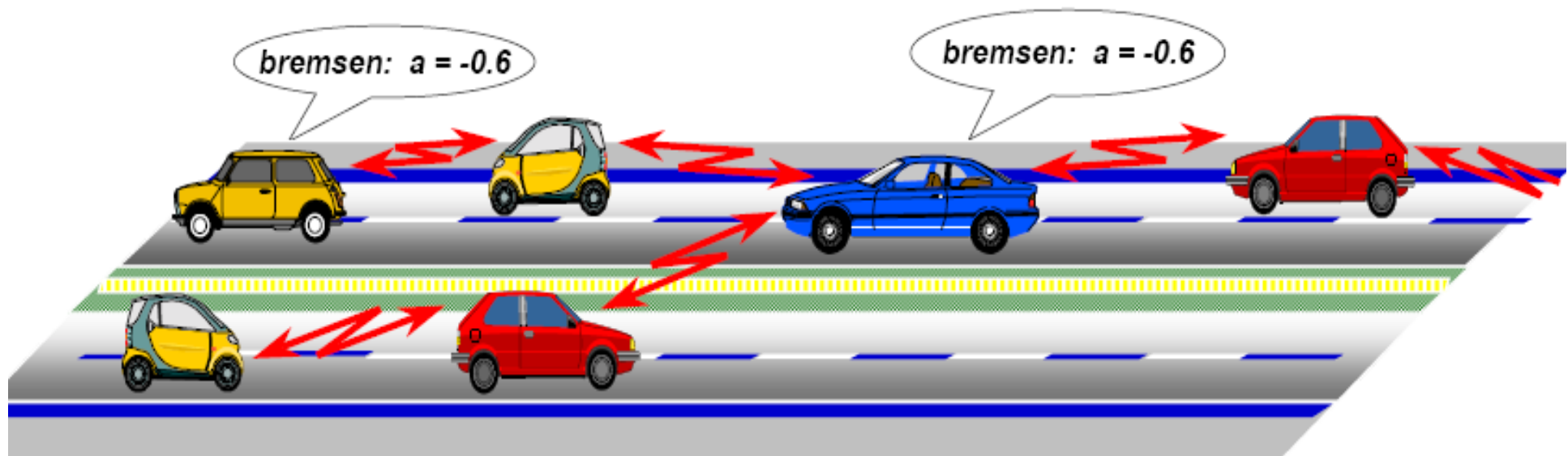
4 Schwerpunktsbereiche:

- Nano science
- Environmental systems research
- Bio Informatics
- Embedded Systems / Transport Technologies

## Automotive Visionen: Sicherheit mit “embedded systems”

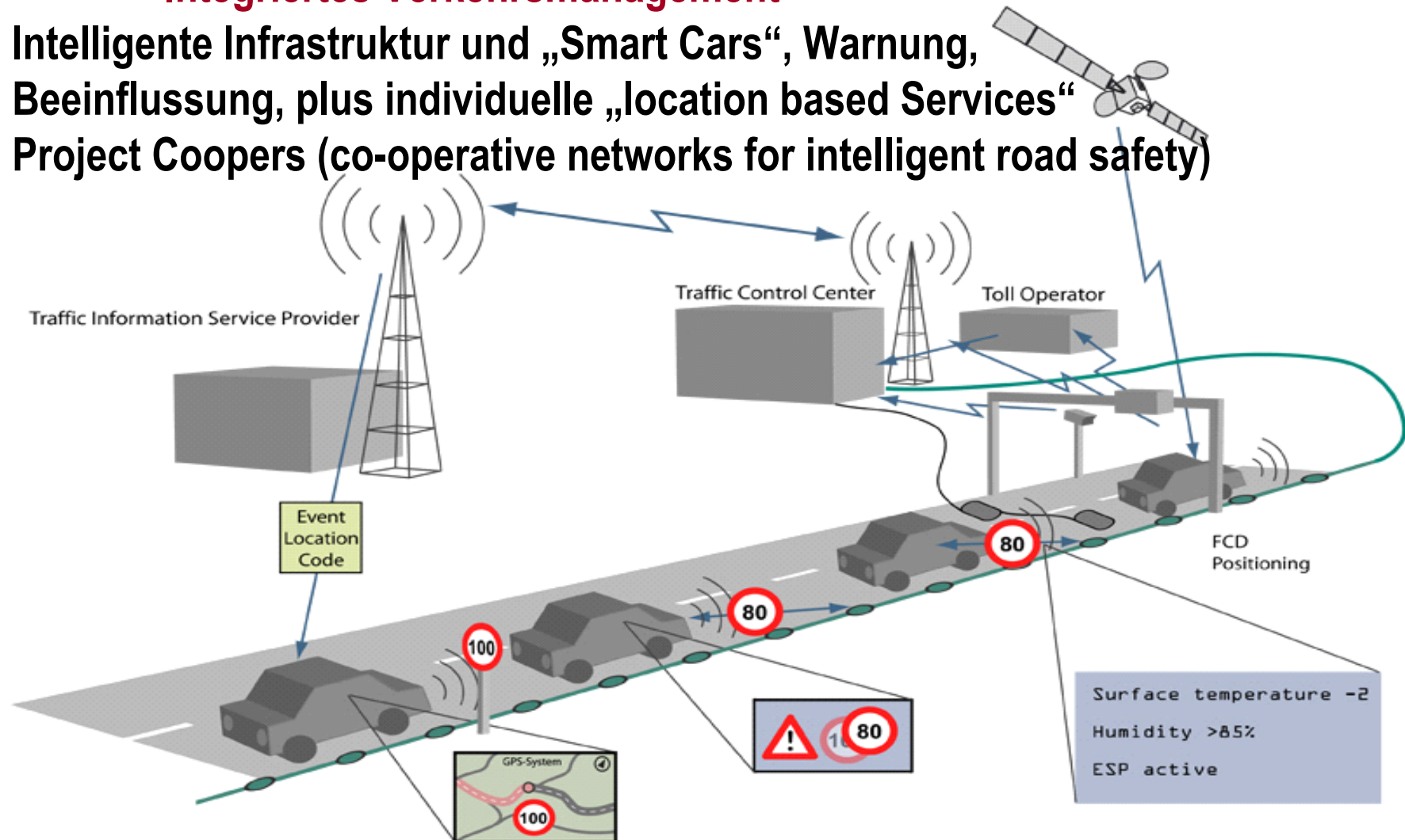
Autos im Geleitzug, Auffahrsicherung, Geisterfahrererkennung, Stauassistent, autonome Fahrzeuge – “elektronische Kopplung” durch Sensorik, Aktorik und sicherheitsrelevante Steuerungselektronik

“Heinzelmännchen” ... oder “...die Geister, die ich rief...”?



## Integriertes Verkehrsmanagement

Intelligente Infrastruktur und „Smart Cars“, Warnung, Beeinflussung, plus individuelle „location based Services“  
Project Coopers (co-operative networks for intelligent road safety)



# STEV ÖSTERREICH

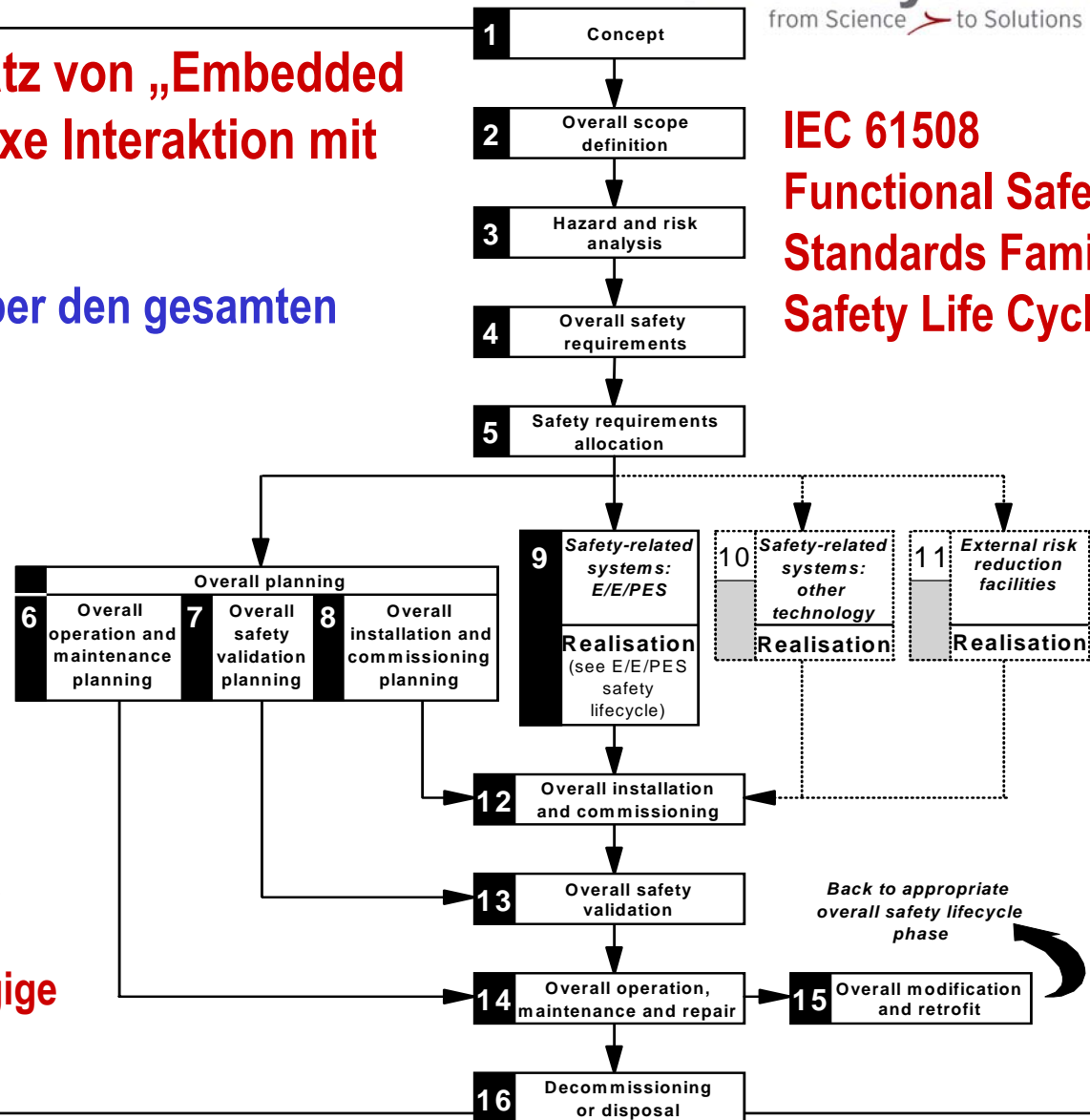
Sicherheitskritischer Einsatz von „Embedded Systems“ – starke, komplexe Interaktion mit Mensch und Umfeld!

Ganzheitliche Systemsicht über den gesamten Lebenszyklus erforderlich:

- Systemkonzept,
- Hazard- und Risikoanalyse,
- Sicherheitsanforderungen,
- Aufteilung auf Systemelemente, Komponenten (HW, SW)

Nachweis der Sicherheit von Konzept und Realisierung her:

- Prüfung weist Konformität mit Standards nach
- Zertifizieren ist die „unabhängige Beglaubigung“!



IEC 61508  
Functional Safety  
Standards Familie  
Safety Life Cycle

## Risikobasierter Ansatz: Welches Risiko ist tolerabel?

- Die Gefährdung, der jemand durch eine technische Einrichtung ausgesetzt ist, muss kleiner sein als die von Natur aus für ihn vorhandene Gefährdung
  - ◆ Mittlere Wahrscheinlichkeit ums Leben zu kommen  $\approx 10^{-2}$  pro Jahr
  - ◆ Sie schwankt sehr mit dem Lebensalter (*Aber: Kennen Sie jemanden, der 150 geworden und gestorben ist? Also genügt es, 150 zu werden!!*)
  - ◆ Sie ist aber nie  $< 10^{-4}$  pro Jahr
  - ◆ Für die Wahrscheinlichkeit, dass jemand durch Versagen einer technischen Einrichtung ums Leben kommt, muss gelten:

$$\underline{p < 10^{-4} \text{ pro Jahr}}$$

Je größer die Zahl der gefährdeten Personen ist, desto sicherer muss die Einrichtung sein.

**High Demand Mode  
(Continuous Mode)**

**– Low Demand Mode  
(max. 1x pro Jahr = 1x pro 10<sup>4</sup> h)**

| SIL | p(Ausfall) pro Stunde                |
|-----|--------------------------------------|
| 4   | $\geq 10^{-9} \text{ .. } < 10^{-8}$ |
| 3   | $\geq 10^{-8} \text{ .. } < 10^{-7}$ |
| 2   | $\geq 10^{-7} \text{ .. } < 10^{-6}$ |
| 1   | $\geq 10^{-6} \text{ .. } < 10^{-5}$ |

| SIL | p(Ausfall) bei Anforderung (PFD)     |
|-----|--------------------------------------|
| 4   | $\geq 10^{-5} \text{ .. } < 10^{-4}$ |
| 3   | $\geq 10^{-4} \text{ .. } < 10^{-3}$ |
| 2   | $\geq 10^{-3} \text{ .. } < 10^{-2}$ |
| 1   | $\geq 10^{-2} \text{ .. } < 10^{-1}$ |

## **IEC 61508 – 3 : Software als Teil des Systems**

### **Allgemeine Anforderungen Software**

- 1. SW-Qualitätsmanagementsystem mit**
  - ◆ **Konfiguration Managementsystem**
  - ◆ **Dokumentation des benutzten Lebenszyklus-/Entwicklungsmodells**
  - ◆ **Dokumentation der SW-Änderungsverfahren**
- 2. unabhängiges Assessment der funktionalen Sicherheit nach der Validierung**
- 3. *Strenge der Anforderungen an Methoden, Werkzeuge, V&V und Assessment durch SIL bestimmt - dafür liefert IEC 61508-3 Vorgaben und Empfehlungen nach SIL! Beispiele auf Folgeseiten***

# IEC 61508-7 C.1: Empfehlungen für Programmiersprachen

| Programming language   | SIL1 | SIL2 | SIL3 | SIL4 |
|--|------|------|------|------|
| 1 ADA  | HR   | HR   | R    | R    |
| 2 ADA with subset  | HR   | HR   | HR   | HR   |
| 3 MODULA-2   | HR   | HR   | R    | R    |
| 4 MODULA-2 with subset   | HR   | HR   | HR   | HR   |
| 5 PASCAL   | HR   | HR   | R    | R    |
| 6 PASCAL with subset   | HR   | HR   | HR   | HR   |
| 7 FORTRAN 77   | R    | R    | R    | R    |
| 8 FORTRAN 77 with subset   | HR   | HR   | HR   | HR   |
| 9 C  | R    | –    | NR   | NR   |
| 10 C with subset and coding standard, and use of static analysis tools | HR   | HR   | HR   | HR   |
| 11 PL/M  | R    | –    | NR   | NR   |
| 12 PL/M with subset and coding standard                                | HR   | R    | R    | R    |
| 13 Assembler   | R    | R    | –    | –    |
| 14 Assembler with subset and coding standard                           | R    | R    | R    | R    |
| 15 Ladder diagrams   | R    | R    | R    | R    |
| 16 Ladder diagram with defined subset of language                      | HR   | HR   | HR   | HR   |

## IEC 61508-3 B.8: Statische Analyse

| Technique/Measure*  |                              | Ref    | SIL1 | SIL2 | SIL3 | SIL4 |
|---|------------------------------|--------|------|------|------|------|
| 1   | Boundary value analysis      | C.5.4  | R    | R    | HR   | HR   |
| 2   | Checklists                   | B.2.5  | R    | R    | R    | R    |
| 3   | Control flow analysis        | C.5.9  | R    | HR   | HR   | HR   |
| 4   | Data flow analysis           | C.5.10 | R    | HR   | HR   | HR   |
| 5   | Error guessing               | C.5.5  | R    | R    | R    | R    |
| 6   | Fagan inspections            | C.5.15 | ---  | R    | R    | HR   |
| 7   | Sneak circuit analysis       | C.5.11 | ---  | ---  | R    | R    |
| 8   | Symbolic execution           | C.5.12 | R    | R    | HR   | HR   |
| 9   | Walk-throughs/design reviews | C.5.16 | HR   | HR   | HR   | HR   |
| *Appropriate techniques/measures shall be selected according to the safety integrity level. |                              |        |      |      |      |      |

## IEC 61508-3 B.2: Dynamische Analyse und Test

| Technique/Measure*  |  | Ref    | SIL1 | SIL2 | SIL3 | SIL4 |
|---|--|--------|------|------|------|------|
| 1   | Test case execution from boundary value analysis | C.5.4  | R    | HR   | HR   | HR   |
| 2   | Test case execution from error guessing          | C.5.5  | R    | R    | R    | R    |
| 3   | Test case execution from error seeding           | C.5.6  | ---  | R    | R    | R    |
| 4   | Performance modelling                            | C.5.20 | R    | R    | R    | HR   |
| 5   | Equivalence classes and input partition testing  | C.5.7  | R    | R    | R    | HR   |
| 6   | Structure-based testing                          | C.5.8  | R    | R    | HR   | HR   |
| <p><b>NOTE – The analysis for the test cases is at the subsystem level and is based on the specification and/or the specification and the code.</b></p> |  |        |      |      |      |      |
| <p><b>* Appropriate techniques/measures shall be selected according to the safety integrity level.</b></p>  |  |        |      |      |      |      |

# Wer soll prüfen? - Was sagt die Norm?

## Unabhängigkeit der Assessoren für funktionale Sicherheit

Betrachtung nach Schadensausmaß oder Safety Integrity Level

| Unabhängigkeits-Level           | Konsequenzen/SIL |                 |                       |           |
|---------------------------------|------------------|-----------------|-----------------------|-----------|
|                                 | A/1              | B/2             | C/3                   | D/4       |
| unabhängige Person              | HR               | HR <sup>1</sup> | NR                    | NR        |
| unabhängige Abteilung           | -                | HR <sup>2</sup> | HR <sup>1</sup>       | NR        |
| <b>unabhängige Organisation</b> | -                | -               | <b>HR<sup>2</sup></b> | <b>HR</b> |

- A...Leichverletzte
- B...ein/wenige Schwerverletzte od.1 Toter
- C...mehrere Tote
- D...zahlreiche Tote

**Akreditiertes Prüflabor = staatlich anerkannte unabhängige Organisation!!**

HR1, HR2 ist applikationsabhängig; eher für HR2 sprechen:

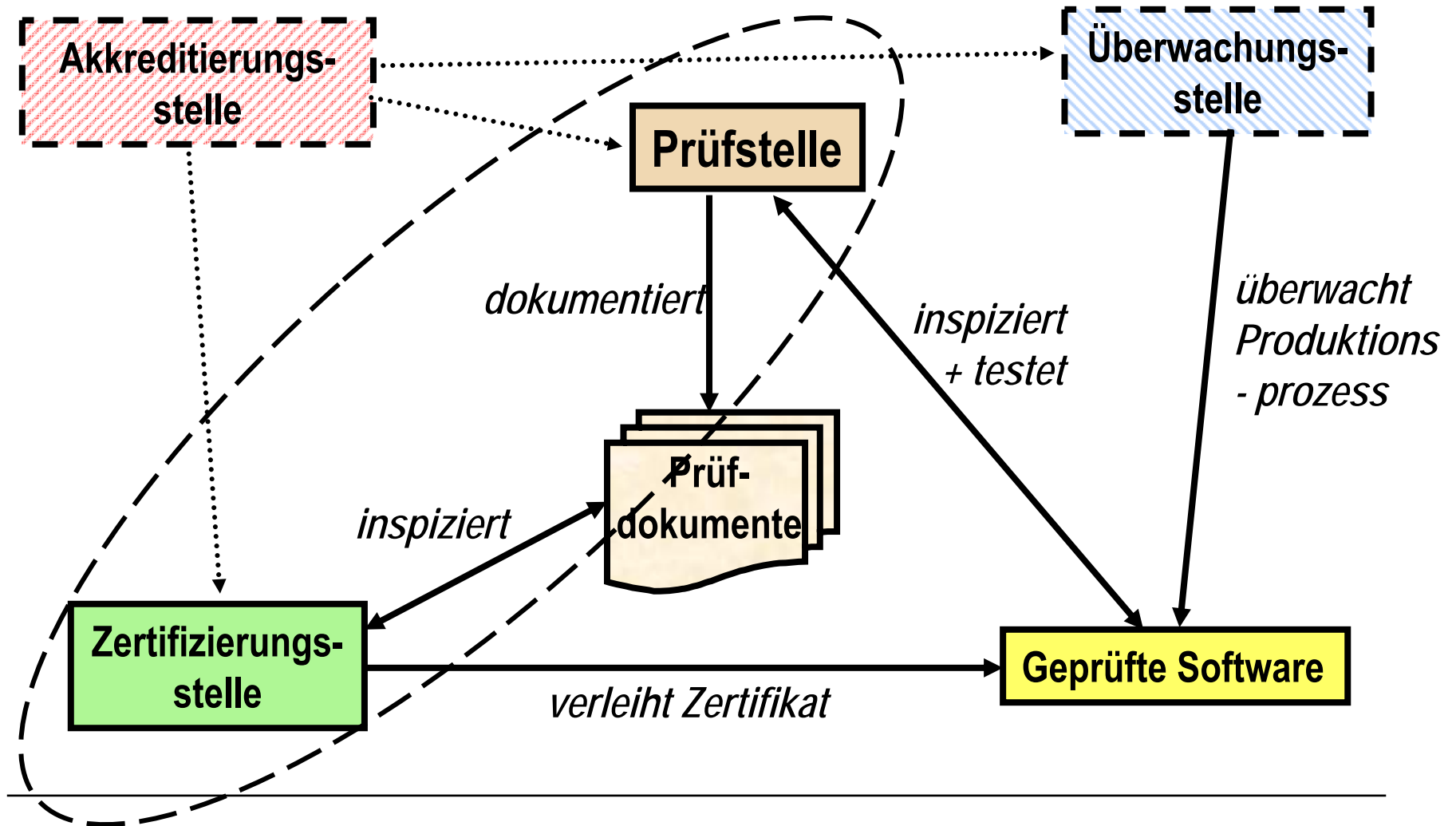
- Projektgröße
- Komplexitätsgrad
- Neuartigkeit der Systemarchitektur
- Neuartigkeit der der Technologie

trifft HR1 zu => dann ist HR2 nicht erforderlich -  
trifft HR2 zu => dann ist HR1 unzureichend,(=NR!)

# Akkreditierte Prüfstelle (Prüflabor)

- Basis Akkreditierungsgesetz (AkkG)
- Zuständig: BMWA
- **Vorteil:** Gilt international – Prüf- und Überwachungsberichte und Zertifizierungen ausländischer Stellen sind „inländischen gleichzuhalten“ wenn Qualifikation der Stelle gleichwertig
- Gilt a priori innerhalb der EU
- Bei Drittstaaten Einhaltung Reziprozitätsprinzip
- **Prüfung = Voraussetzung für die Zertifizierung**  
→ liefert Unterlagen für Zertifizierungsstelle

# Prüfstelle – Zertifizierungsstelle - Überwachungsstelle



## **Voraussetzungen und Pflichten einer Prüfstelle**

- **Fachlich Ausgebildeter je Prüfgebiet**
- **Beaufsichtigung des Prüf-Personals durch qualifizierte Personen**
- **Personal frei von kommerziellem, finanziellem od. sonstigem Einfluss**
- **Keine Abhängigkeit der Vergütung des Prüfpersonal von**
  - ◆ **Zahl der Prüfungen**
  - ◆ **Ergebnis der Prüfungen**
- **Verantwortung, Befugnisse und Wechselbeziehungen d.MA definieren**
- **Stellvertreter des leitenden Personals festlegen**
- **Vertraulichkeit gewährleisten bzgl. Kundendaten**
- **QM-System + unabhängigen Qualitätsmanager haben**

## **Akkreditierte Prüfstelle für Software: V&V Lab ARC**

Relevante internationale Standards:

- **Prüflabor nach ISO/IEC 17025 (für alle Prüflabors)**  
**General requirements for the competence of testing and calibration laboratories**
- **SW-Sicherheitskriterien nach ISO/IEC 61508**  
**Functional Safety of Electrical/Electronic/Programmable Systems**
- **Software Evaluierung nach ISO/IEC 9126-1**  
**Software Engineering – Product Quality**
- **Akkreditierung derzeit: Unit Testing, kann bei Bedarf auf andere Verfahren leicht ausgedehnt werden.**

## **Gesamtes Leistungsangebote ARC**

- **Akkreditiertes Prüflabor, unabhängiges Testen**
- **Vom ASQF-zerifizierte Tester**
- **CheckLab: Prüfung der Spezifikationen – Fokus -> Requirements**
- **Qualifiziertes Testtool zur Automatisierung und zum Regressionstest**
- **Dynamisches und Strukturelles Testen**
- **TestLab: Systematischer Unittests**
- **Integrationstests**
- **RAMS-Analysen (Systemisch: Hardware, Software, Humanware)**
  - ◆ **FMECA - Failure Modes and Effects Criticality Analysis**
  - ◆ **FTA - Fault Tree Analysis**
  - ◆ **HAZOP - Hazard and Operability Analysis**

## Erfahrungen des V&V-Labors

### Warum Schwerpunkt Unit-Testing?

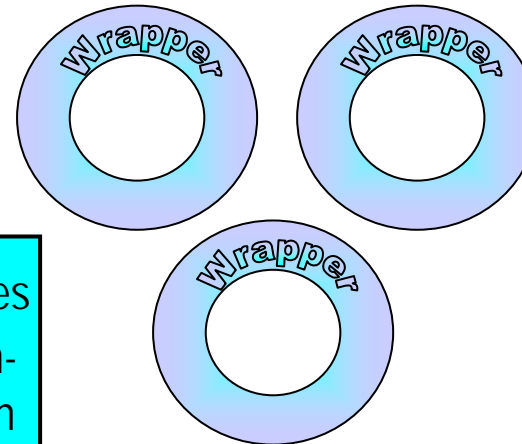
- + **Ideal für modulare (o-o) Software, gut eingeführte Technik**
- + **Wird als die effektivste Methode zum Testen individueller Softwarekomponenten angesehen, um Verhalten an Grenzwerten und Code-Abdeckung zu testen\* (ESA)**
- **Ziemlich großer Aufwand – aber ideal für Automatisierung**
- **Deckt Integrationsfragen nicht ab (dafür andere Testtools vorhanden)**

\* Ellims, M., Bridges, J., Ince, D. C.: Unit Testing in Practice (ISSRE'04)

---

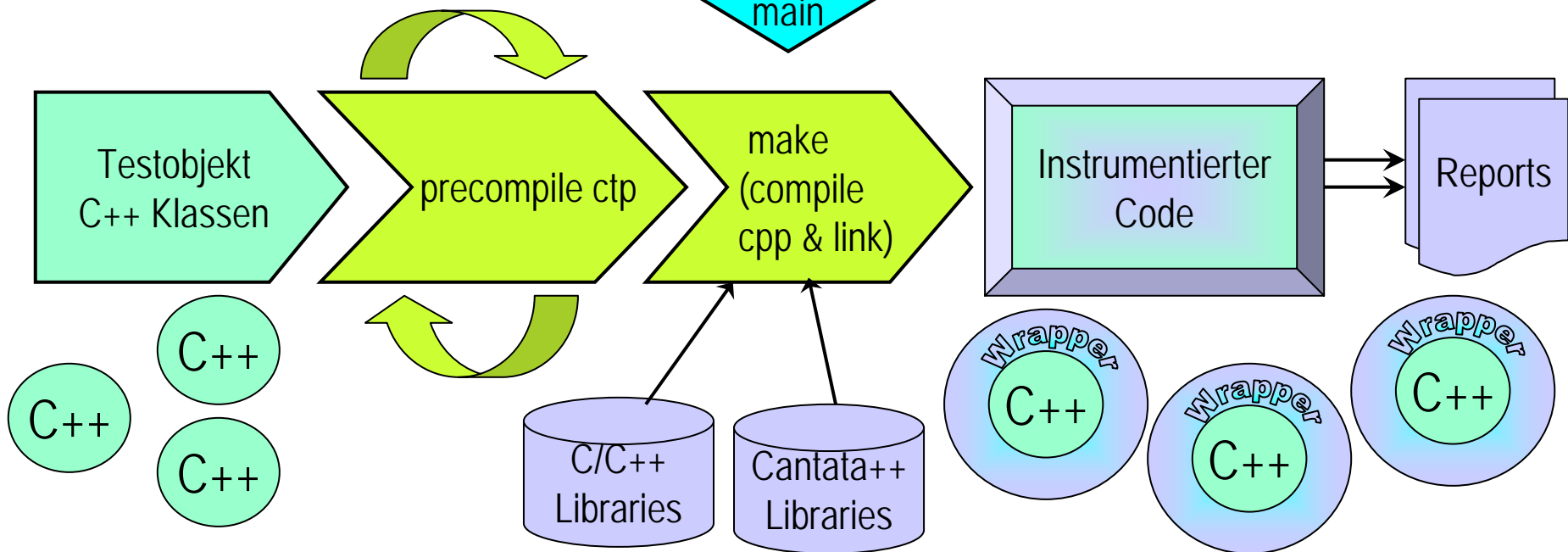
# Unit Test mit IPL Cantata++ im Detail

Coverage,  
White & Black  
Box Testfälle



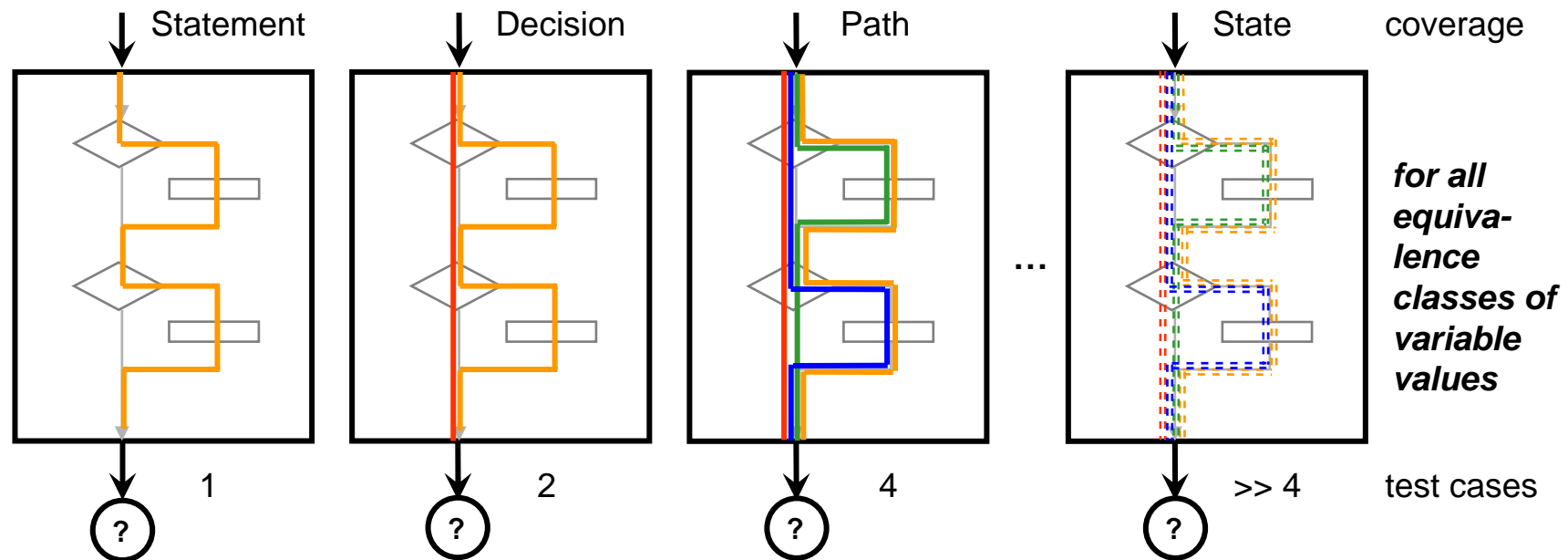
Cantata++  
Wrapper  
Klassen

C++ Frames  
mit Prüfan-  
weisungen  
main



# Testabdeckung: Cantata++ unterstützt viele Metriken

- Wie findet man möglichst viele Fehler?
- Indem man möglichst jedes Softwarestück einmal ausführt → **coverage**



Ideal 100%, meist <100% wegen „toten Codes“ (-), „defensive programming“ (+)...  
 ... und: auch 100% coverage deckt nicht alle Fehler auf (abgesehen von State Coverage)

## Testendekriterium

*„Wenn alles durchgetestet ist ....“*

- **Maximale Restfehlerzahl**
- **Kurve gefundener Fehler pro Zeiteinheit verflacht**

### *Black Box Tests*

- Funktionalität komplett
- Fehlerkurve verflacht
- Neue Testfälle bei Re-Test

### *White Box Tests*

- ~100% Testdeckungsgrad
- Fehlerkurve verflacht
- Neue Testfälle bei Re-Test

## Ergebnisse / 1

### ■ Aufwand:

#### ◆ 40.000 Minuten (~67 min. pro Methode)

- Entwurf und Implementierung von Testfällen
- Testdurchführung und Verfeinerung der Testfälle, um die Abnahmekriterien zu erfüllen (z.B. Mindestcoverage)
- Configuration Management für mehrere hundert Objekte
- Dokumentation  
(z.B. Begründung z.B. bei Nicht-Erreichenkönnen der geforderten Coverage)
- Für jeden gefundenen Fehler: Software Problem Reporting mit internem Tool
- Qualitätssicherungsmaßnahmen (z.B.. Testfälle an Hand von Checklisten überüprüfen)

### ■ Erzielte Coverage: 97.7%

- ◆ „defensive programming“ (e.g. unreachable 'default' in case-stmt.)

### ■ Gefundene Fehler: > 100

---

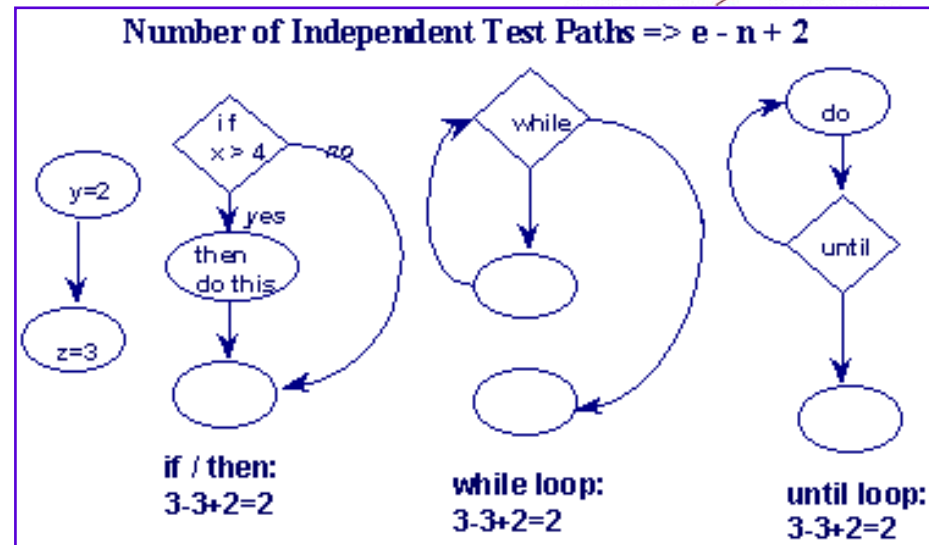
## Ergebnisse / 2 – Gruppen von Mängeln

- **(49%) Documentation faults**
  - ◆ (54%) Value never returned
  - ◆ (31%) Returned value not documented
  - ◆ (10%) Documentation fault e.g. incorrect struct description
  - ◆ (5%) Documentation not sufficient
- **(33%) Coding faults**
  - ◆ (38%) Implementation dependencies e.g. not always initialized vars
  - ◆ (23%) Wrong pointer handling e.g. NULL ptr dereferencing
  - ◆ (13%) Inversion of 'true' and 'false' 0 = true, 1 = false
  - ◆ (26%) Other
- **(17%) Incomplete coverage**
  - ◆ (65%) Defensive programming always true assertions
  - ◆ (35%) Default or else branch unreachable branches
- **(1%) Other faults**

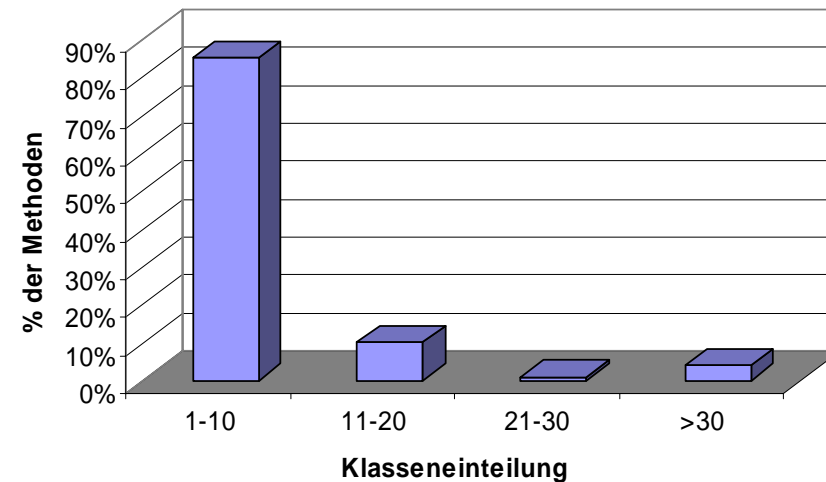
# Zyklomatische Komplexität

$$c = e - n + 2$$

c = Zyklomatische Komplexität  
 e = Anzahl der Kanten (edge)  
 n = Anzahl der Knoten (node)



Histogramm - Zyklomatische Komplexität



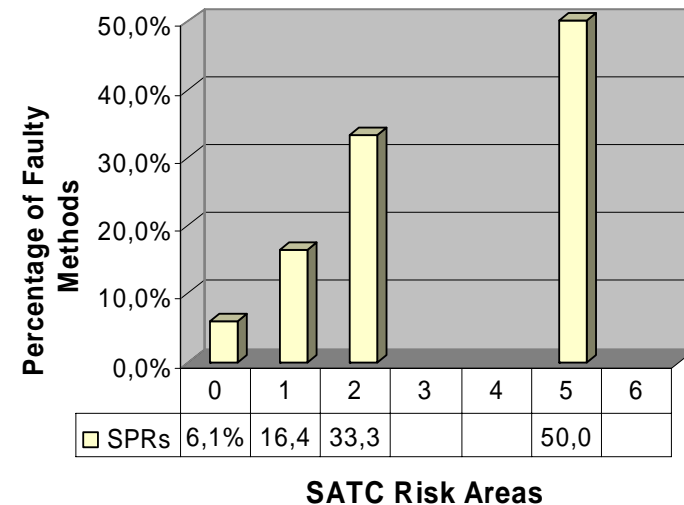
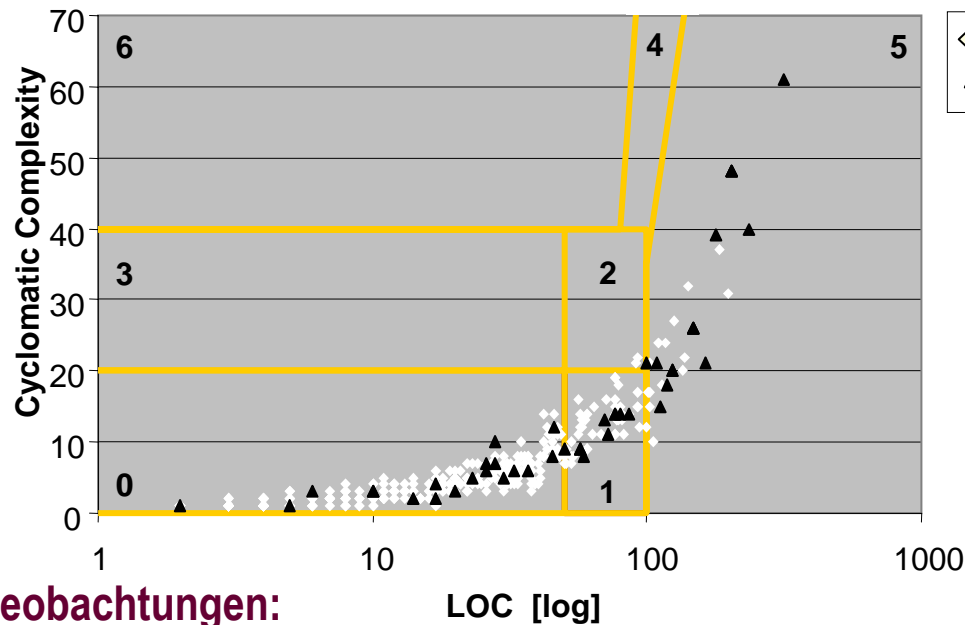
Typ. Literaturangabe (unabh. von LOC):

| Zykl. Kompl. | Risiko                                  |
|--------------|---|
| 1-10         | einfaches risikoarmes Programm          |
| 11-20        | komplexer, mäßiges Risiko               |
| 21-50        | komplex hohes Risiko                    |
| >50          | unstabiles Programm (sehr hohes Risiko) |

## Vergleiche: SPR vs. CC, using NASA SATC's "Risk Areas" \*

Analyse Größe (LOC) vs. zyklomat. Komplexität und Fehleranzahl (SPRs) der Methoden

SATC: Je höher die Risikobereichszahl, desto höher ist das Risiko von Fehlern



### Beobachtungen:

- ◆ schmalbandig
- ◆ offenkundig gleichmäßige Verteilung fehlerhafter Methoden über das gesamte Band  $\Rightarrow$  keine Korrelation mit dem Risikobereich?

\* SATC: Software Assurance Test Center

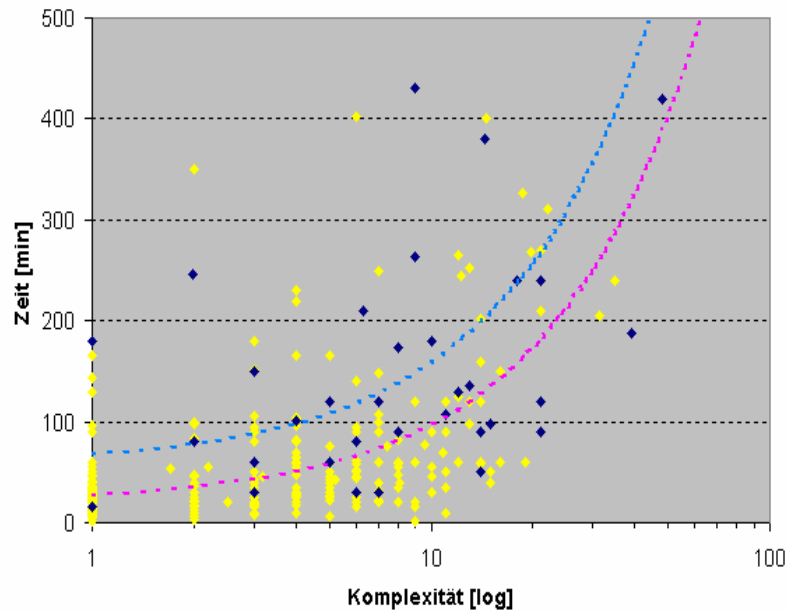
CC: Cyclomatic Complexity

SPR: Software Problem Report

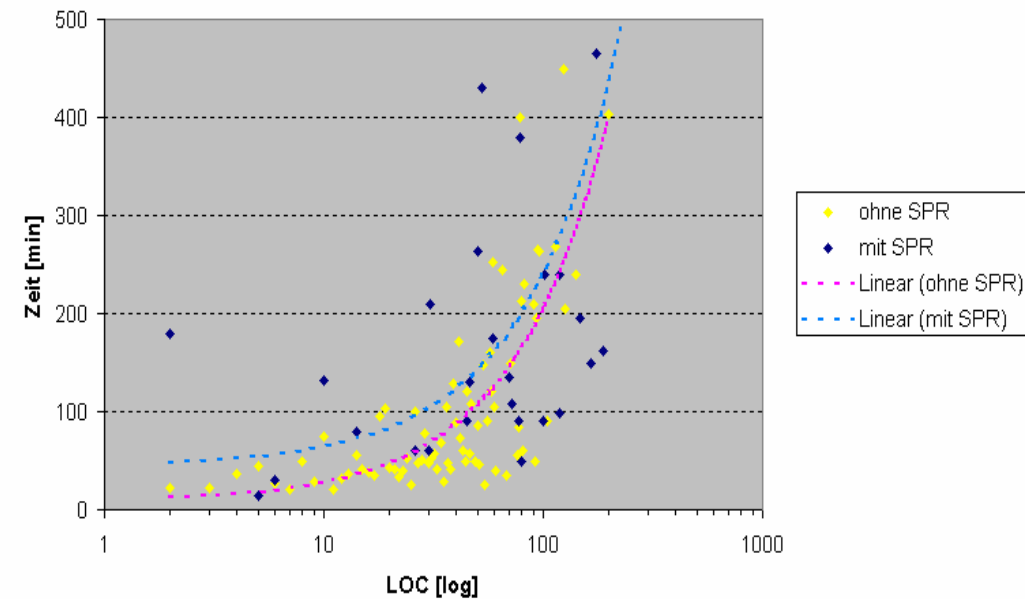
# Auswirkung Komplexität/LOC → Bearbeitungszeit

Testaufwandsanalyse bzgl. Komplexität und LOC

Zusammenhang Komplexität - Bearbeitungszeit



Zusammenhang LOC - Bearbeitungszeit



## Beobachtungen:

- Ungefähr linearer Zusammenhang (LOC und CC)
- Bearbeitungszeit ca. linear zu Anzahl SPRs (Software Problem Reports)
- Auch Methoden einer Klasse mit CC=1 ohne SPR benötigen bis zu 3 Stunden

## Resultate (klassebezogen nach Klassenfehlerrate):

- **Normierung**

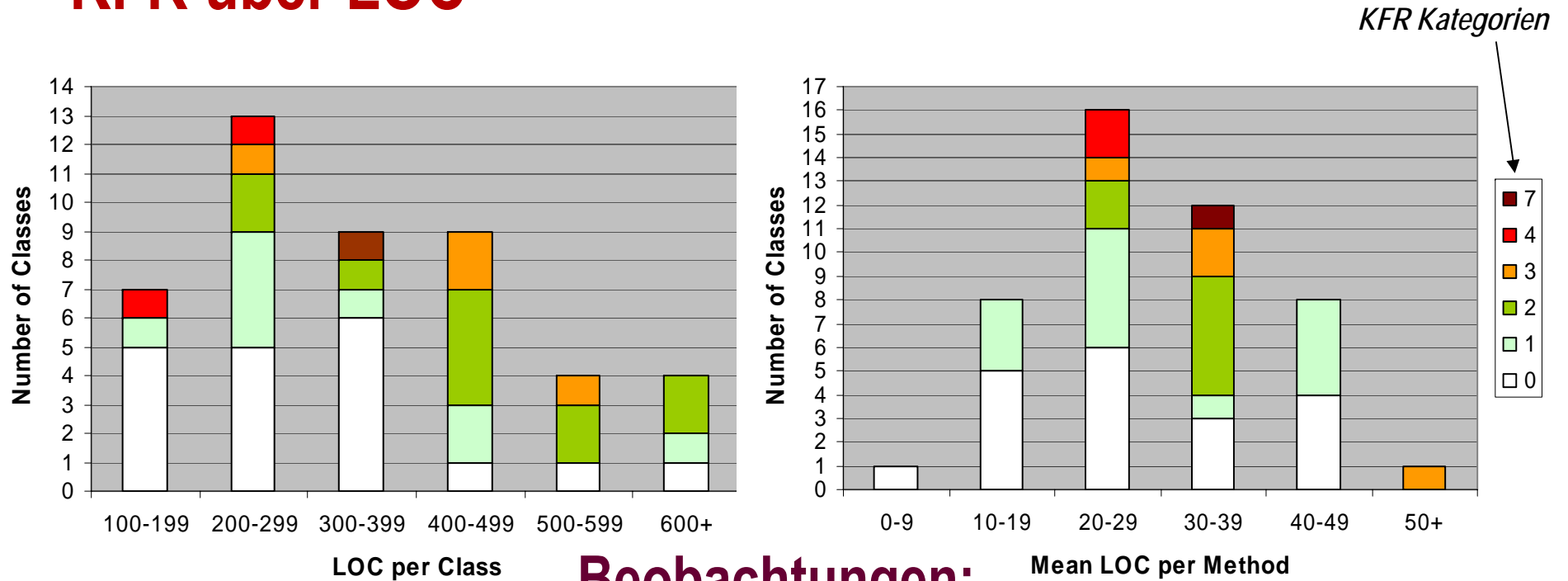
- ◆ *Klassen-Fehler-Rate*  $KFR = |SPR| / |M|$  pro Klasse (Class Failure Rate CFR)  
 $|SPR|$  = Anzahl der SPRs pro Klasse,  
 $|M|$  = Anzahl der Methoden pro Klasse

- **Kategorisierung nach KFR-Bereichen**  
 (einige kleinere Klassen nicht berücksichtigt)

| Risiko-Kategorie | KFR-Bereich | Anz. Klassen |
|------------------|-------------|--------------|
| 0                | 0.0         | 19           |
| 1                | (0.0, 0.1]  | 9            |
| 2                | (0.1, 0.2]  | 11           |
| 3                | (0.2, 0.3]  | 4            |
| 4                | (0.3, 0.4]  | 2            |
| 5                | (0.4, 0.5]  | 0            |
| 6                | (0.5, 0.6]  | 0            |
| 7                | (0.6, 0.7]  | 1            |

(SPR = Software Problem Report)

# KFR über LOC

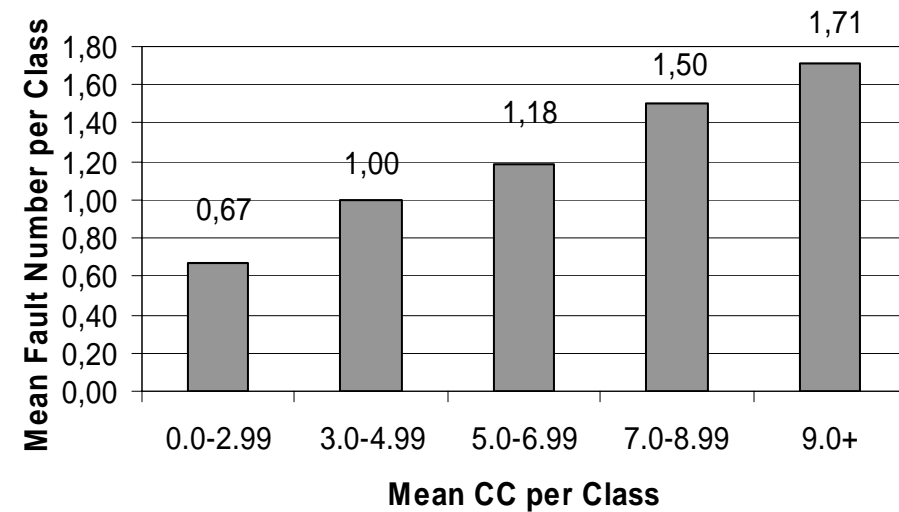
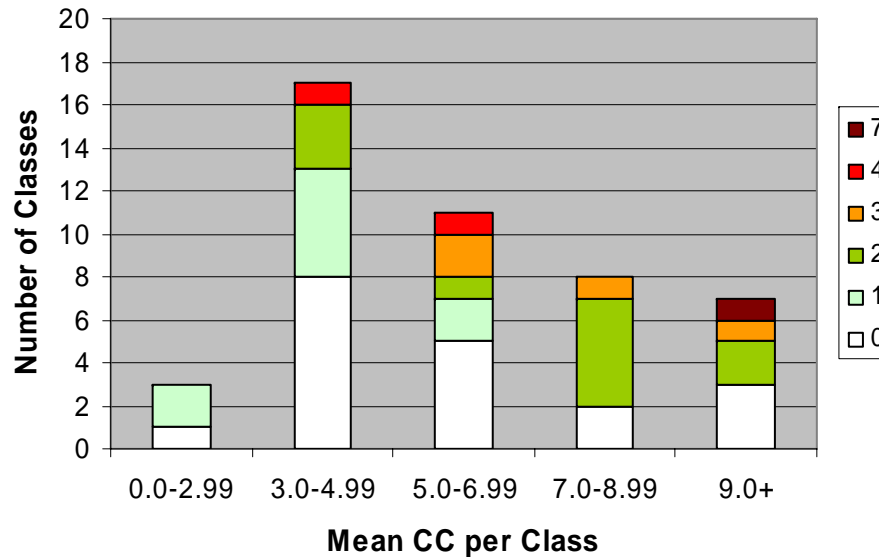


## Beobachtungen:

➤ Zusammenhang zwischen „Risikohöhe der Klassen“ zur LOC/class ratio and umgekehrt ist eher schwach

➤ „Normalisierung“ auf Methoden zeigt mehr direkte Korrelation

# KFR über mittlerer zyklomat. Komplexität CC pro Klasse

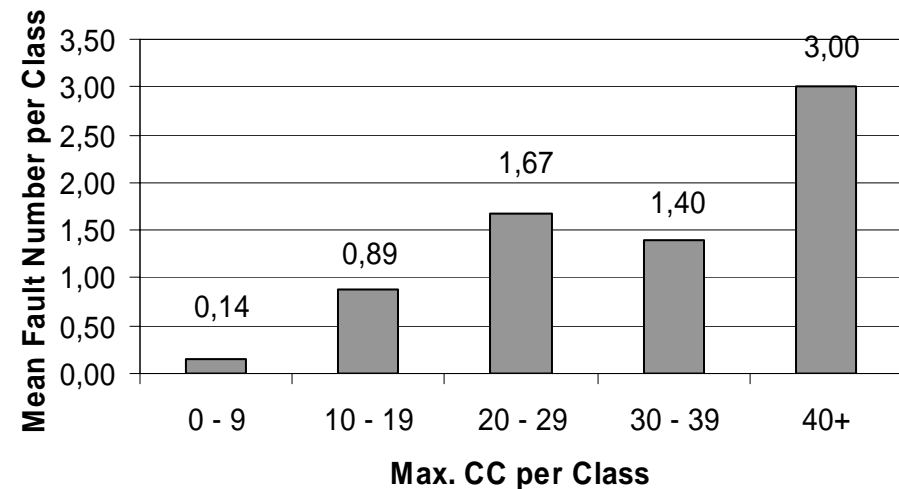
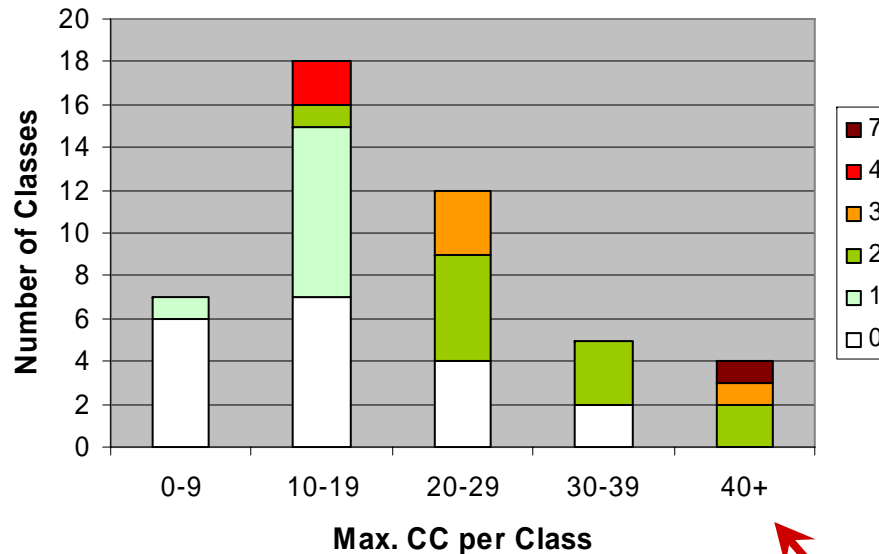


## Beobachtungen:

➤ Klassen mit KFR=0 gibt es in allen Kategorien, aber High Risk“ Klassen zeigen eher höhere mittlere CC der Methoden pro Klasse

➤ Anzahl der Fehler in einer Gruppe gleicher mittlerer CC, dividiert durch Anzahl der Klassen in dieser Kategorie (linear steigend)

# KFR über maximaler zyklomat. Komplexität CC pro Klasse



## Beobachtungen:

- Auch bei Kategorisierung nach dem im Vergleich zum Mittelwert einfacheren Maß des Maximums der CC jeder Klasse bleibt die Korrelation sichtbar, wenn auch schwächer.

## Schlussfolgerungen:

- **LOC ist ganz allgemein kein guter Risikoindikator**
  - ◆ Stimmt gut mit der Literatur überein, z.B. Fenton & Ohlsson, 2000: *no support for "size metrics (such as LOC) are good predictors of number of pre-release faults in a module"*
- **CC ist klar besser**
  - ◆ Stimmt gut mit der Literatur überein, z.B. Ohlsson & Alberg, 1996: *"simple complexity measures seem to be as useful as more complicated measures"*
- **Gilt auch bei Betrachtung von Klassen**
- *Je höher die mittlere (maximale) CC einer (C++) Klasse ist, desto höher ist das Risiko von Fehlern in der ganzen Klasse.*

# *Danke für Ihr Interesse !*

**Dipl.-Ing. Thomas Gruber**  
**Austrian Research Centers GmbH - ARC**  
**GmbH**  
**Smart Systems**  
**Donau City Straße 1**

**Telefon: +43 (0) 50 550 - 4106**  
**Fax: +43 (0) 50 550 – 4150**  
**Email: [thomas.gruber@arcs.ac.at](mailto:thomas.gruber@arcs.ac.at)**  
**Web: [www.smart-systems.at](http://www.smart-systems.at)**

**Dipl.-Ing. Erwin Schoitsch**  
**Austrian Research Centers GmbH - ARC**  
**GmbH**  
**Smart Systems**  
**Donau City Straße 1**

**Telefon: +43 (0) 50 550 - 4117**  
**Fax: +43 (0) 50 550 – 4190**  
**Email: [erwin.schoitsch@arcs.ac.at](mailto:erwin.schoitsch@arcs.ac.at)**  
**Web: [www.smart-systems.at](http://www.smart-systems.at)**