HEICON
Global Engineering

Vollständigkeit und Konsistenz von Requirements durch non-intrusive Systembeobachtung

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

HEICON – Global Engineering GmbH
www.heicon-ulm.de | blog.heicon-ulm.de

Martin Heininger

- ✓ Studium der Elektrotechnik an der FH Augsburg
- ✓ Geschäftsführer Heicon – Global Engineering GmbH
- ✓ 20 Jahre Projekterfahrung in der Luftfahrt
- ✓ 10 Jahre Projekterfahrung in Automotive, Bahn und Industrie
- ✓ Projektberatung und Hands-On Coaching mit folgenden Schwerpunkten
  - Requirements Engineering
  - Test Engineering
  - Management der Funktionalen Sicherheit

© HEICON – Global Engineering GmbH

HEICON
Global Engineering

- Sicherheitsprojekte – Kulturprägende Aspekte
- Stand der Technik: Vollständigkeit mit struktureller Source Code Coverage
- Stand der Technik: Vollständigkeit durch Nachweis von Daten-/Kontrollflüssen
- Neue Optionen durch die non-intrusive Systembeobachtung
- Fazit und Ausblick

© HEICON – Global Engineering GmbH

HEICON
Global Engineering

Sicherheitsprojekte – Kulturprägende Aspekte

Stand der Technik: Vollständigkeit mit struktureller Source Code Coverage

Stand der Technik: Vollständigkeit durch Nachweis von Daten-/Kontrollflüssen

Neue Optionen durch die non-intrusive Systembeobachtung

Fazit und Ausblick

© HEICON – Global Engineering GmbH

Sicherheitsprojekte – Kultur prägende Aspekte

Normen/Gesetze

Abnahmebehörden

ISO26262 - DO178C – IEC61508 – ISO13849 – ISO25119

© HEICON – Global Engineering GmbH

Sicherheitsprojekte – Kultur prägende Aspekte

HEICON
Global Engineering

Fast alle Normen der Funktionalen Sicherheit lassen sich in 4 Themengebiete einteilen:

System Engineering: Entwurf von Systemen von denen keine Gefahr ausgeht!

Hardware Engineering:
Minimierung zufälliger Fehlern durch Bauteilqualität und Redundanzkonzepte

Software Engineering:
Minimierung systematischer Fehlern durch Prozesse und Methoden (z.B. Requirements, Tests etc.)

Management der Funktionalen Sicherheit: Sicherheitskultur

© HEICON – Global Engineering GmbH

Sicherheitsprojekte – Kultur prägende Aspekte

HEICON Global Engineering

Stärken und Schwächen von Elementen des Software Entwicklungsprozesses

Requirements		Funktionalität
		Vollständigkeit
Dynamische Tests		Funktionalität
		Vollständigkeit
Strukturelle Coverage		Funktionalität
		Vollständigkeit (Mathematisch beweisbar!)

© HEICON – Global Engineering GmbH

Sicherheitsprojekte – Kultur prägende Aspekte

Nachweis der Vollständigkeit ist ein absolutes „MUST HAVE“:

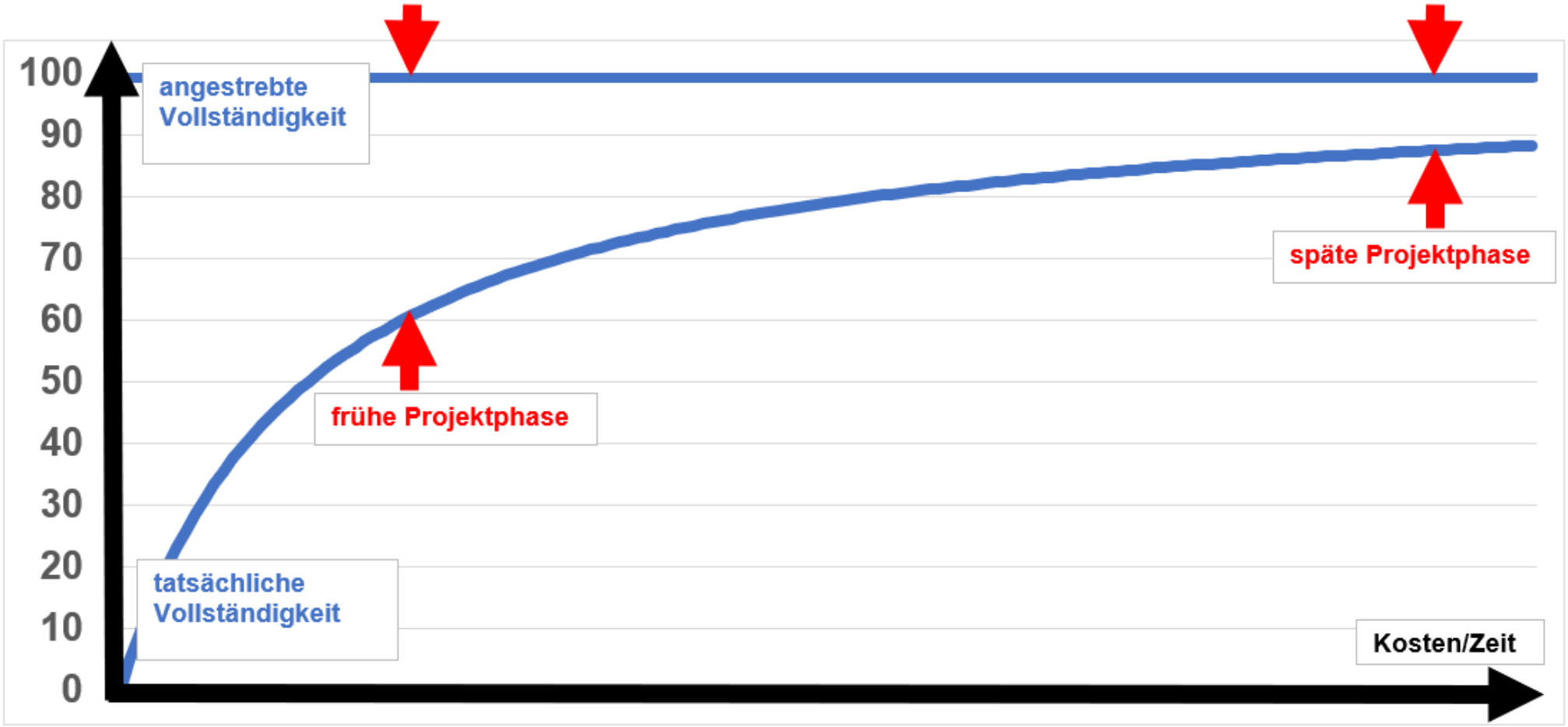
- Requirements
- Architektur
- Tests

© HEICON – Global Engineering GmbH

Sicherheitsprojekte – Kultur prägende Aspekte

HEICON
Global Engineering

Vollständigkeit von Requirements, Architektur und Tests:



100% Vollständigkeit gibt es nicht! Stattdessen, Minimierung der Lücke!

© HEICON – Global Engineering GmbH

HEICON
Global Engineering

Sicherheitsprojekte – Kulturprägende Aspekte und Definition

Stand der Technik: Vollständigkeit mit struktureller Source Code Coverage

Stand der Technik: Vollständigkeit durch Nachweis von Daten-/Kontrollflüssen

Neue Optionen durch die non-intrusive Systembeobachtung

Fazit und Ausblick

© HEICON – Global Engineering GmbH

Strukturelle Source Code Coverage – Stand der Technik

Strukturelle Tests versus Funktionale Tests

Strukturelle Tests: Nachweis der Ausführung der internen Struktur von Komponenten.

```
if A or (B and C) then
   do_something;
else
  do_something_else
end if;
```

Funktionale Tests: Nachweis, dass die Black-Box Anforderungen getestet sind.

```
Die Heizungssteuerung soll 2
Minuten nach dem die
Temperatur unter 5°C
gefallen ist, das
Ausgangssignal Vent2 auf 5
Volt setzen.
```

© HEICON – Global Engineering GmbH

Strukturelle Source Code Coverage – Stand der Technik

HEICON Global Engineering

Strukturelle Tests: Statement Coverage

```
if A or (B and C) then
    do_something;
else
  do_something_else
end if;
```

Nr.	A	B	C	Condition	Decision
1	False	False	False	False	do_something_else
2	False	False	True	False	do_something_else
3	False	True	False	False	do_something_else
4	False	True	True	True	do_something
5	True	False	False	True	do_something
6	True	False	True	True	do_something
7	True	True	False	True	do_something
8	True	True	True	True	do_something

$$C_0 = \frac{\text{Anzahl der ausgeführten Statements}}{\text{Gesamtzahl aller Statements}}$$

Test Cases für 100%
Statement Coverage: 2 und 4

© HEICON – Global Engineering GmbH

Strukturelle Source Code Coverage – Stand der Technik

HEICON Global Engineering

Strukturelle Tests: Modified Condition /Decision Coverage

Beweise das jede Variable der Bedingung einen Einfluss aufs Ergebnis hat

```
if A or (B and C) then
   do_something;
else
  do_something_else
end if;
```

Test Cases für 100% MC/DC Coverage: 2,3,4 and 6

Nr.	A	B	C	Condition	Decision
2	False	False	True	False	do_something_else
6	True	False	True	True	do_something

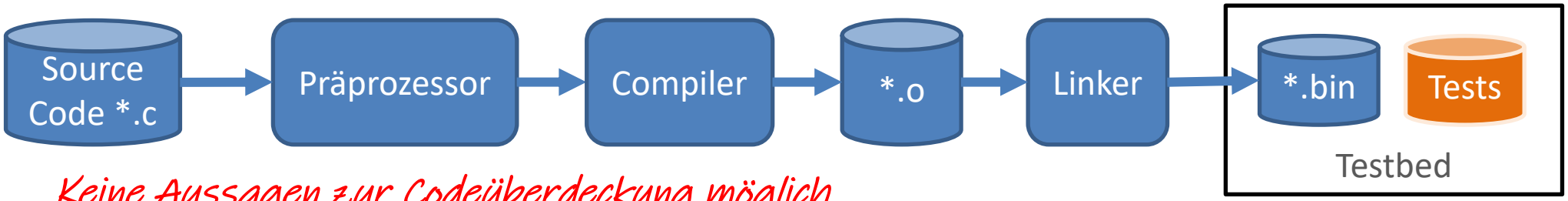
Nr.	A	B	C	Condition	Decision
2	False	False	True	False	do_something_else
4	False	True	True	True	do_something

Nr.	A	B	C	Condition	Decision
3	False	True	False	False	do_something_else
4	False	True	True	True	do_something

© HEICON – Global Engineering GmbH

Strukturelle Source Code Coverage – Stand der Technik

Bisherige Technologie zur Messung der strukturellen Coverage:

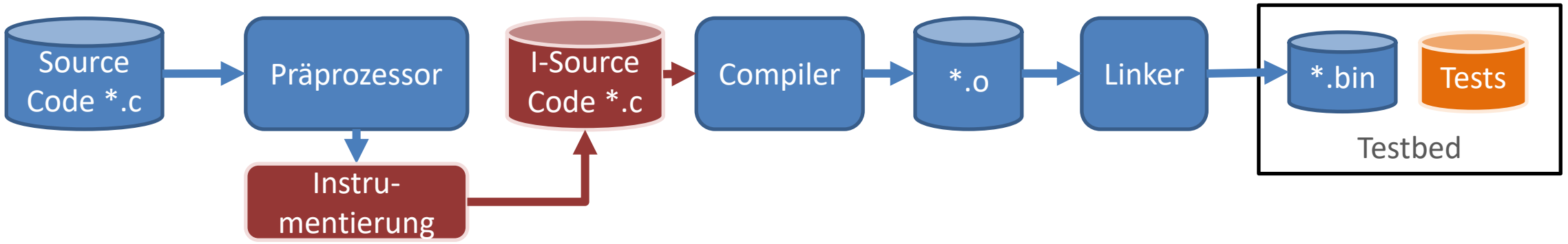


Keine Aussagen zur Codeüberdeckung möglich

Instrumentierung

```
if (a<0)
   { /* … */}
else
   { /* … */}
```

```
if ((a<0)?(zaehler1++,1):(zaehler2++,0))
   { /* … */}
else
   { /* … */}
```



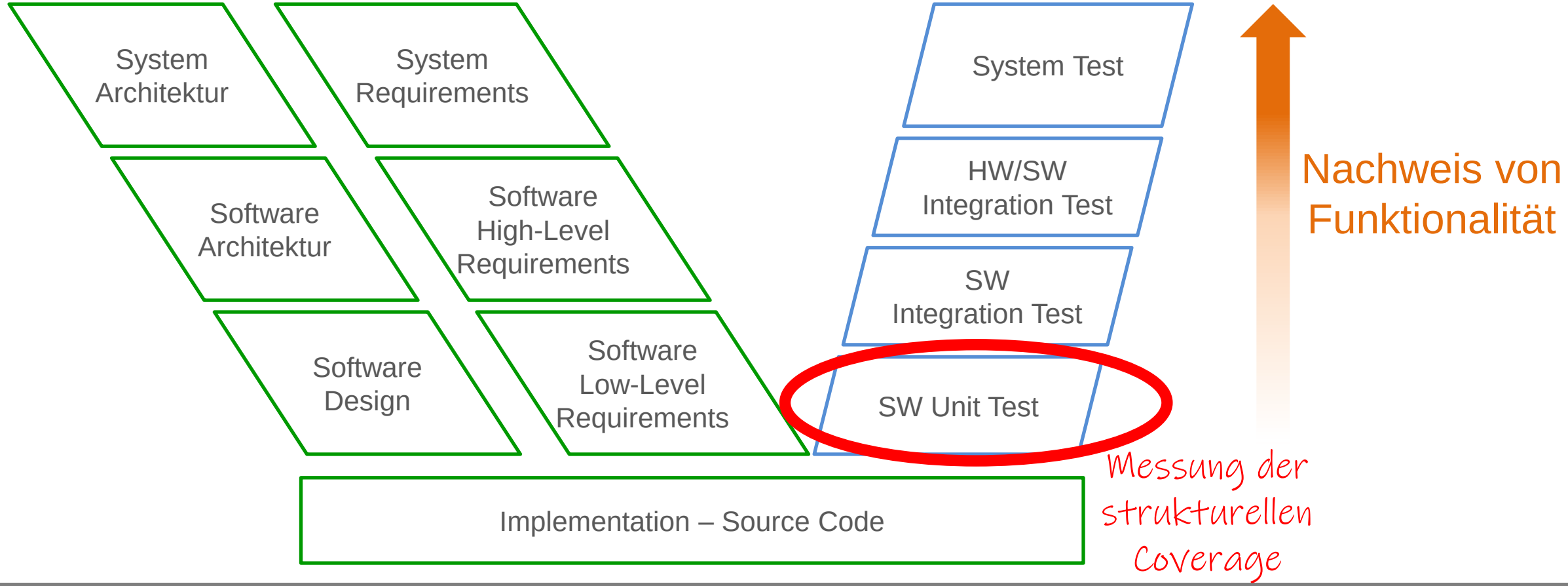
© HEICON – Global Engineering GmbH

Strukturelle Source Code Coverage – Stand der Technik

Nachteile der Instrumentierung: Veränderung des Laufzeitverhaltens des Systems

➔ Messung der Coverage nur auf Unittest-Ebene sinnvoll möglich

➔ Kein Test von funktionalen Software oder System Requirements auf Unittestebene

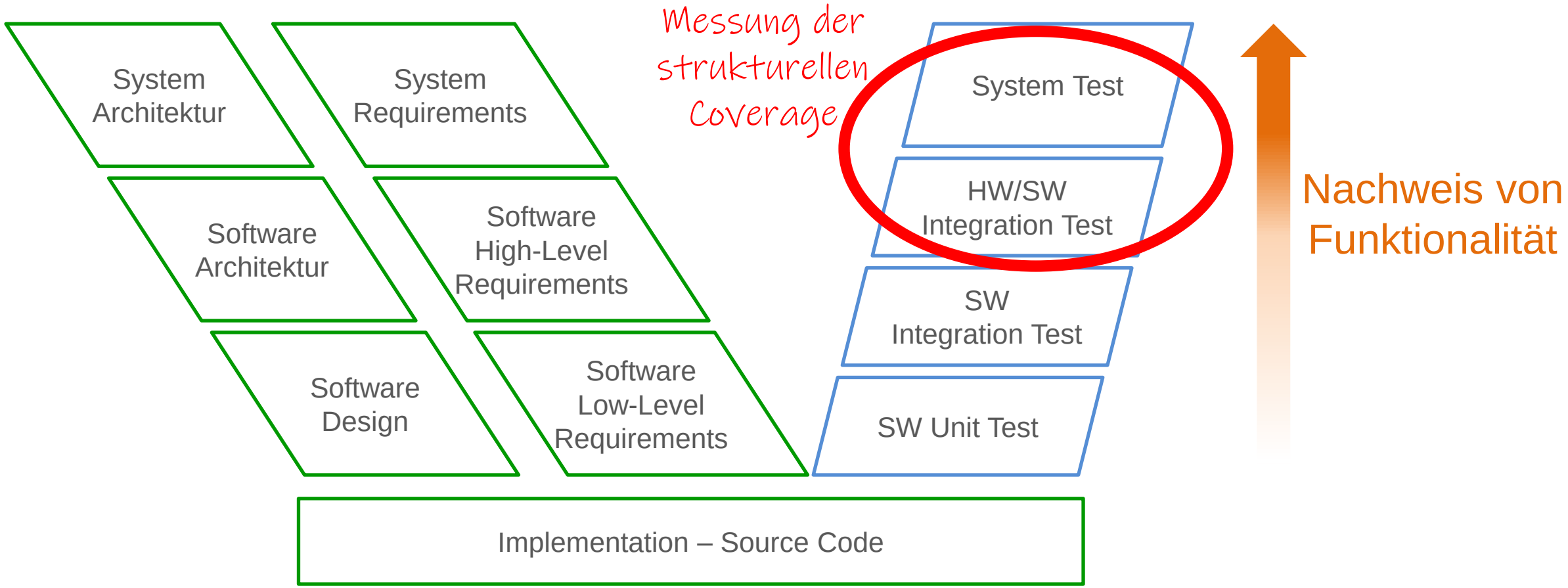


© HEICON – Global Engineering GmbH

Strukturelle Source Code Coverage – Stand der Technik

Ursprüngliche Idee zur Messung der strukturellen Coverage:

Strukturellen Coverage wird als Teil von **funktionalen**, Requirements basierten Tests bestimmt, um die Stärken und Schwächen der Methoden auszugleichen.



HEICON
Global Engineering

Sicherheitsprojekte – Kulturprägende Aspekte

Stand der Technik: Vollständigkeit mit struktureller Source Code Coverage

Stand der Technik: Vollständigkeit durch Nachweis von Daten-/Kontrollflüssen

Neue Optionen durch die non-intrusive Systembeobachtung

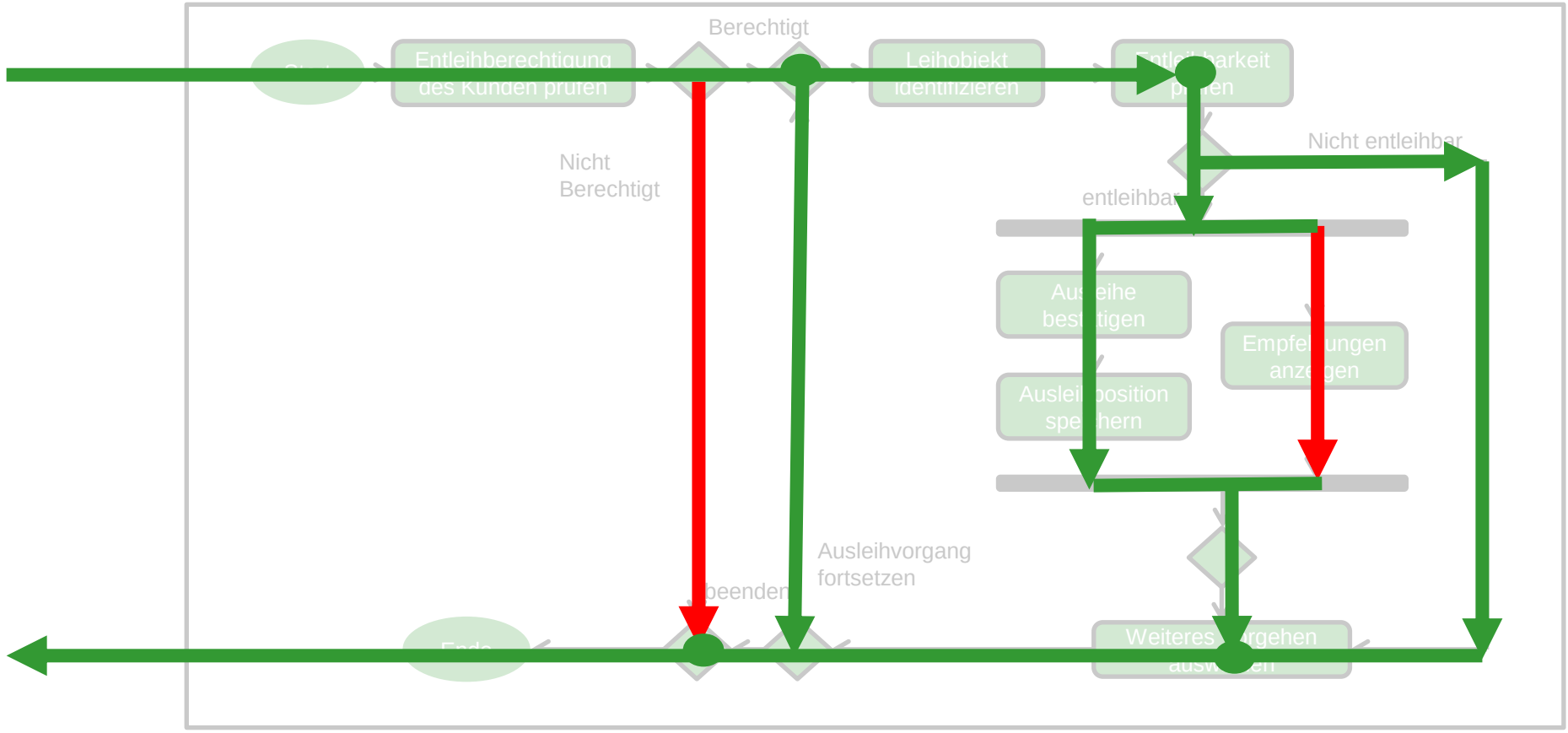
Fazit und Ausblick

© HEICON – Global Engineering GmbH

Daten-/Kontrollflüsse – Stand der Technik

HEICON Global Engineering

Daten- und Kontrollflüsse:



… werden in der Software Architektur und/oder im Modell definiert!

Daten-/Kontrollflüsse – Stand der Technik

Daten- und Kontrollflüsse:

Daten-/Kontrollflüsse – Stand der Technik

- ✓ Funktionale Sicherheitsstandards wie ISO 26262 (Automobil) und DO 178C (Luftfahrt) fordern eine Daten- und Kontrollflussanalyse
- ✓ Die Spezifikation von Daten- und Kontrollflüssen in der Architektur wird stark vernachlässigt
- ✓ Eine bewusste Traceability zwischen funktionalen Requirements und dem jeweils dazugehörigen Daten- und Kontrollfluss findet nicht statt
- ✓ Bisher ist nur eine statische Analyse von komplexeren Daten- und Kontrollflüssen in einer Software durchführbar
- ✓ Die DO 178C fordert explizit den Nachweis von **Daten und Kontrollflusskopplung** durch **Requirement basierte Tests**!
- ✓ Dynamisch Testen kann man Software Daten- und Kontrollflüsse nicht auf höheren Testebenen, da hierfür auch eine Instrumentierung oder ein projektspezifische Schnittstelle notwendig wäre

Mit bisheriger Technologie ist kein systematischer, dynamischer Nachweis möglich

© HEICON – Global Engineering GmbH

HEICON
Global Engineering

Sicherheitsprojekte – Kulturprägende Aspekte

Stand der Technik: Vollständigkeit mit struktureller Source Code Coverage

Stand der Technik: Vollständigkeit durch Nachweis von Daten-/Kontrollflüssen

Neue Optionen durch die non-intrusive Systembeobachtung

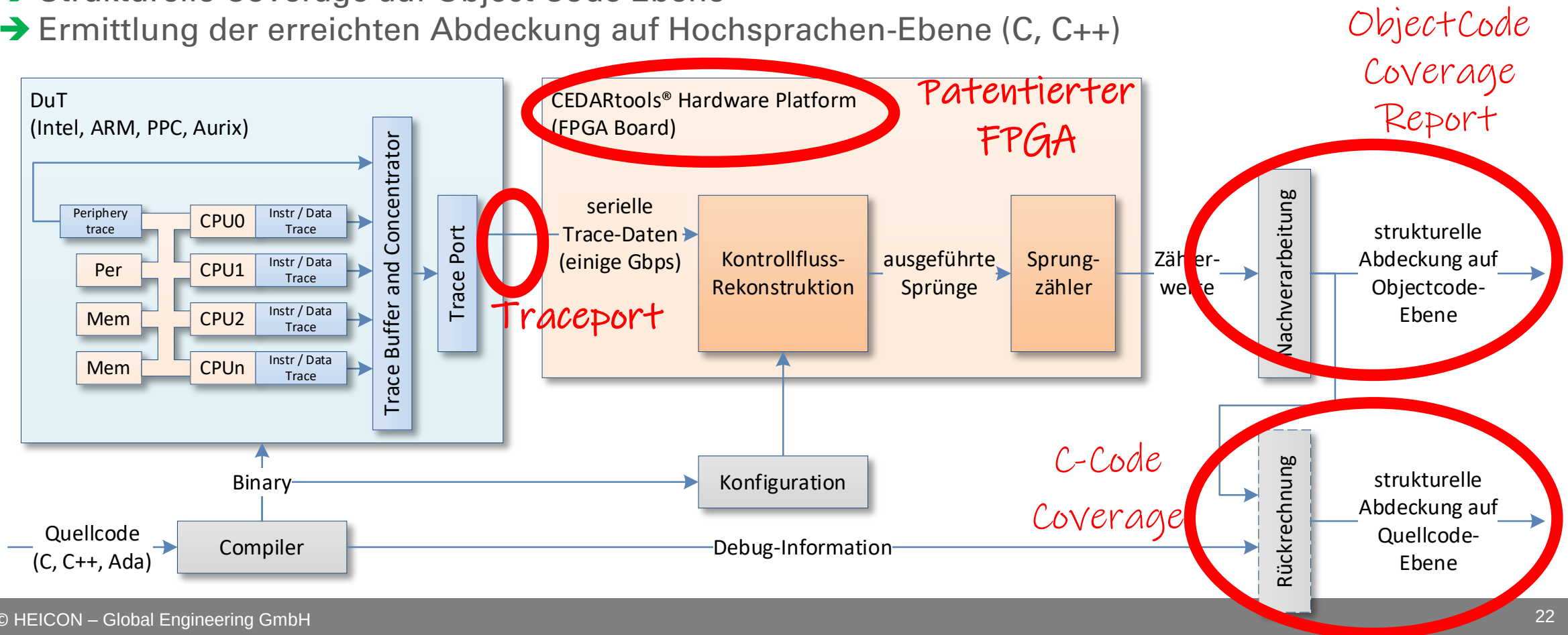
Fazit und Ausblick

© HEICON – Global Engineering GmbH

Non-Intrusive Systembeobachtung

Schlüsselelemente:

- ➔ Beobachtung des Systems über die Traceschnittstelle
- ➔ Zeitliche unbegrenzte parallele, live Verarbeitung der Daten mittel patentierter HW
- ➔ Strukturelle Coverage auf Object Code Ebene
- ➔ Ermittlung der erreichten Abdeckung auf Hochsprachen-Ebene (C, C++)



Non-Intrusive Systembeobachtung

Mapping der Object Code Coverage auf die Coverage auf Hochsprachenebene (C, C++)

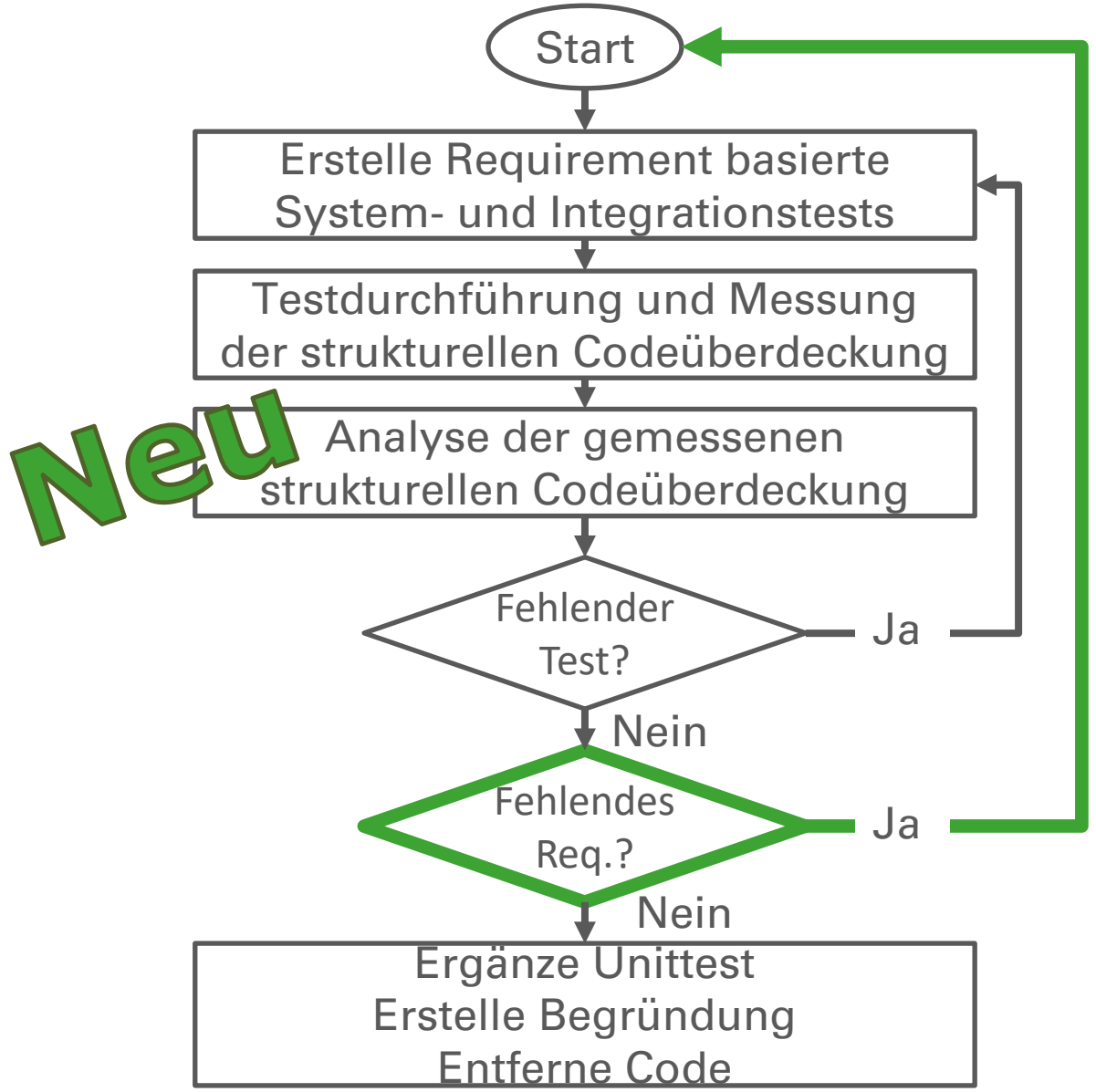
```
     |  |     0040090f:    mov     qword ptr [rbp - 8], rax
7    |  |     00400913:    mov     rax, qword ptr [rbp - 8]
7    |  |     00400917:    mov     qword ptr [rax + 8], 0
7    |  |     0040091f:    mov     rax, qword ptr [rbp - 8]
7    |  |     00400923:    mov     rdx, qword ptr [rax + 8]
7    |  |     00400927:    mov     rax, qword ptr [rbp - 8]
7    |  |     0040092b:    mov     qword ptr [rax + 0x10], rdx
7    |  |     0040092f:    mov     rax, qword ptr [rbp - 8]
7    |  |     00400933:    mov     edx, dword ptr [rbp - 0x1c]
7    |  |     00400936:    mov     dword ptr [rax], edx
7    |  |     00400938:    mov     rax, qword ptr [rbp - 0x18]
7    |  |     0040093c:    mov     rdx, qword ptr [rbp - 8]
7    |  |     00400940:    mov     qword ptr [rax], rdx
     |  7     00400943:    jmp     0x400993
10   |  |     00400945:    mov     rax, qword ptr [rbp - 0x18]
10   |  |     00400949:    mov     rax, qword ptr [rax]
10   |  |     0040094c:    mov     eax, dword ptr [rax]
10   |  |     0040094e:    cmp     dword ptr [rbp - 0x1c], eax
     |  5     00400951:    jge     0x40096d
5    |  |     00400953:    mov     rax, qword ptr [rbp - 0x18]
5    |  |     00400957:    mov     rax, qword ptr [rax]
5    |  |     0040095a:    lea     rdx, [rax + 0x10]
5    |  |     0040095e:    mov     eax, dword ptr [rbp - 0x1c]
5    |  |     00400961:    mov     esi, eax
5    |  |     00400963:    mov     rdi, rdx
     |  5     00400966:    call    0x4008e2
     |  5     0040096b:    jmp     0x400993
5    |  |     0040096d:    mov     rax, qword ptr [rbp - 0x18]
5    |  |     00400971:    mov     rax, qword ptr [rax]
5    |  |     00400974:    mov     eax, dword ptr [rax]
5    |  |     00400976:    cmp     dword ptr [rbp - 0x1c], eax
```

Line	Branch	Instr.	Count	Source
1				#ifdef CEDAR
2				#include "cedar_ipt.h"
3				#endif
4				
5				#include <stdlib.h>
6				#include <stdio.h>
7				
8				struct bin_tree {
9				int data;
10				struct bin_tree * right, *left;
11				};
12				typedef struct bin_tree node;
13				
14				void insert(node ** tree, int val)
15		100%	17x	{
16		100%	17x	node *temp = NULL;
17	2/2	100%	17x	if (!(*tree))
18				{
19		100%	7x	temp = (node *)malloc(sizeof(node));
20		100%	7x	temp->left = temp->right = NULL;
21		100%	7x	temp->data = val;
22		100%	7x	*tree = temp;
23		100%	7x	return;
24				}
25				
26	2/2	100%	10x	if (val < (*tree)->data)
27				{
28		100%	5x	insert(&(*tree)->left, val);
29				}

© HEICON – Global Engineering GmbH

Neue Optionen: Non-Intrusive Systembeobachtung

HEICON Global Engineering



Neue Optionen: Non-Intrusive Systembeobachtung

HEICON Global Engineering

Zwischenergebnisse aus dem KMU-Innovative Forschungsprojekt CoCoSI
Accemic, Heicon, Intel, Fraunhofer IESE:

- ✓ Wenn Fault-Injection Tests gemacht werden, sind 80% und mehr an Statement Coverage auf Systemebene realistisch
- ✓ Um eine sinnvolle Aussage der Coverage Messung zu bekommen, muss eine strenge Kopplung zwischen Testausführung und Coverage Messung erfolgen
- ✓ Der Unittest kann von formalen Anforderungen entlastet werden
- ✓ **Ansatz unterstützt die modellbasierte Entwicklung bei der angestrebt wird, nur noch eine Ebene von textuellen Requirements zu haben**

Test Result: Statement Coverage

```
if A or (B and C) then
   do_something;
else
  do_something_else
end if;
Switch (Select)
Case ONE:
   do_something_if _Select_one;
   break;
Case TWO:
   do_something_if _Select_two;
   break;
```

Statement covered

Statement not covered

Neue Optionen: Non-Intrusive Systembeobachtung

HEICON Global Engineering

Zwischenergebnisse aus dem KMU-Innovative Forschungsprojekt CoCoSI
Accemic, Heicon, Intel, Fraunhofer IESE:

Welchen Szenarien tragen wesentlich zu Code Coveragelücken aus Systemebene bei:

- ✓ Wirtschaftlichkeit der Testumgebung
  - Test von funktionalen Verhalten außerhalb von Labortemperaturen
  - Simulationen für alle externen Fehlerfälle
- ✓ Gefährdung des Testpersonals durch Sensoren / Aktuatoren
  - Sehr schnell drehende Teile, z.B. Motoren
  - Messung von (hohen) Gaskonzentrationen
- ✓ Robustheit der entwickelten Software
  - Verhalten bei mechanisch verriegelten Kombinationen von Sensorwerten, z.B. hohe Geschwindigkeit und gleichzeitig extremer Lenkwinkel
  - Defensiv Programmierter Code
  - Lesbarkeit von Source Code

Test Result: Statement Coverage

```
if A or (B and C) then
   do_something;
else
  do_something_else
end if;
Switch (Select)
Case ONE:
   do_something_if _Select_one;
   break;
Case TWO:
   do_something_if _Select_two;
   break;
```

Statement covered

Statement not covered

Neue Optionen: Non-Intrusive Systembeobachtung

HEICON Global Engineering

Zwischenergebnisse aus dem KMU-Innovative Forschungsprojekt CoCoSI
Accemic, Heicon, Intel, Fraunhofer IESE:

ACCEMIC TECHNOLOGIES

File Navigation	Line coverage	Branch coverage
- ..	92.80%	74.00%
- ..	92.80%	74.00%
- RollerCoaster	98.41%	90.91%
RollerCoaster.cpp	98.41%	90.91%
- RollerCoasterLib	90.91%	69.23%
DiscountEngine.cpp	80.77%	64.29%
PricingEngine.cpp	94.34%	50.00%
TicketApp.cpp	94.81%	71.43%
TicketCategory.cpp	86.67%	90.00%

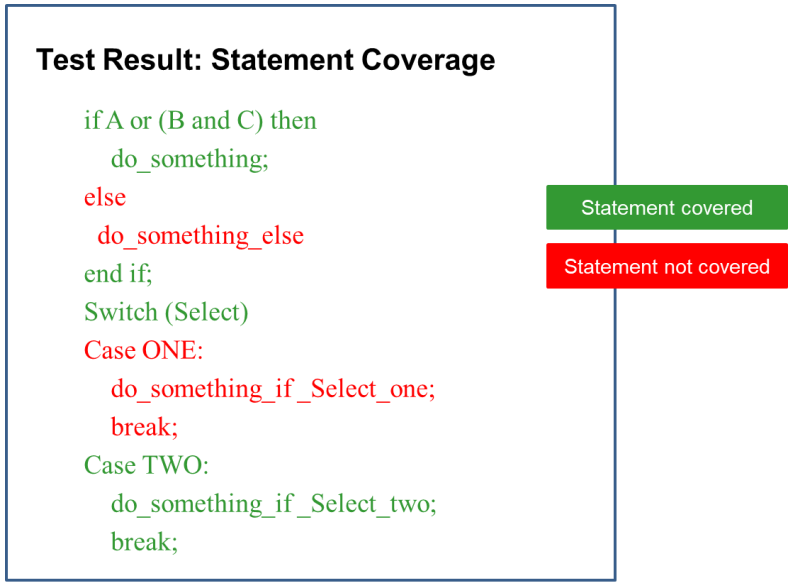
Hohe Branch-Coverage mit **Systemtests**, trotz Laborbeispiel, welches optimiert wurde möglichst "schlechte" Werte zu liefern

Neue Optionen: Non-Intrusive Systembeobachtung

Zwischenergebnisse aus dem KMU-Innovative Forschungsprojekt CoCoSI
Accemic, Heicon, Intel, Fraunhofer IESE:

Aktuell noch in Bearbeitung:

- ✓ Coverage Messungen in industriellem Projekt
- ✓ Implementierung der MC/DC
- ✓ Analyse der „neuen“ ObjectCode basierten Coveragelücken, wie z.B. Grenzwertüberprüfungen und zusätzliche Sprungbedingungen aufgrund von Codeoptimierungen



© HEICON – Global Engineering GmbH

HEICON
Global Engineering

Sicherheitsprojekte – Kulturprägende Aspekte

Stand der Technik: Vollständigkeit mit struktureller Source Code Coverage

Stand der Technik: Vollständigkeit durch Nachweis von Daten-/Kontrollflüssen

Neue Optionen durch die non-intrusive Systembeobachtung

Fazit und Ausblick

© HEICON – Global Engineering GmbH

Fazit und Ausblick

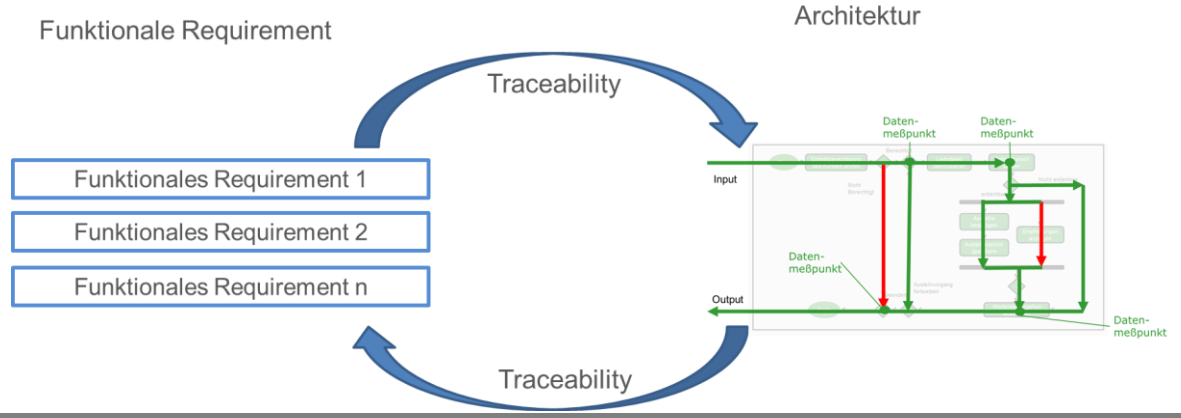
Fazit:

Non-intrusive Systembeobachtungs-Technologie ermöglicht es zukünftig die ursprüngliche Motivation zur Verwendung strukturelle Coverage Messung umsetzen zu können:

- ✓ Unterstützung des Requirement Engineerings durch Vollständigkeitsprüfung
- ✓ Identifikation von fehlenden Testfällen im Falles fehlender Code Coverage

Ausblick:

Non-intrusive Systembeobachtungs-Technologie kann zukünftig den Nachweis des korrekten Daten- und Kontrollflusses einer Software durch Tests unterstützen. Bisher werden Daten- und Kontrollflüsse fast ausschließlich per statische Analyse nachgewiesen.



© HEICON – Global Engineering GmbH

Förderung

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS19044C gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Die Partner im Verbundprojekt CoCoSI waren:

- ✓ Intel Deutschland GmbH, Neubiberg
- ✓ Accemic Technologies GmbH, Kiefersfelden
- ✓ Heicon – Global Engineering GmbH, Schwendi
- ✓ Fraunhofer-Gesellschaft, IESE Kaiserslautern

© HEICON – Global Engineering GmbH

HEICON
Global Engineering

HEICON
Global Engineering

Wenn Sie Fragen haben zu

- ✓ Teststrategien
- ✓ Requirements Engineering
- ✓ Funktionaler Sicherheit
- ✓ Entwicklungsprozessen

rufen Sie uns an: **0176 24 73 99 60** oder schreiben Sie uns eine Mail: **info@heicon-ulm.de**

Kontakt

Martin Heininger

Dipl.-Ing (FH)

Tel.: +49 7353 981781

Büros

Kreuzweg 22 | 88477 Schwendi

In der Neuen Welt 8 | 87700 Memmingen