Evaluierung von Software-Verifikationswerkzeugen

Verbreitete Irrglauben zu Software-Verifikationswerkzeugen

R. Gerlich[1], R. Gerlich[1], C.R. Prause[2]

ESE-Kongress 2016

01.12.2016, Sindelfingen

[1] Dr. Rainer Gerlich BSSE System and Software Engineering
Immenstaad, Germany
E-Mail: Rainer.Gerlich@bsse.biz
Ralf.Gerlich@bsse.biz

[2] Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Bonn, Germany
E-Mail: Christian.Prause@dlr.de

Inhalt

- Einführung
- Projekt
- Lessons Learned
- Ergebnisse
- Zusammenfassung und Ausblick

Einführung

Vorbemerkung 1

BSSE System and Software Engineering

DLR

Die hier präsentierten Ergebnisse **hängen stark** von folgenden Randbedingungen ab:

- der Fokus liegt auf Sicherheit (Safety)

  Filterung der Werkzeug-Meldungen auf Fehler, die den Betrieb einschränken oder “Best Practices” verletzen

- die analysierte Software hat ein spezielles Fehlerprofil

  einige Fehlertypen sind möglicherweise in der Software nicht vertreten

- nur eine Untermenge an Funktionen konnte im Detail evaluiert werden

  einige Fehlertypen sind möglicherweise in dieser Untermenge nicht vertreten

- die Anzahl von Fehlern eines Fehlertyps geht in das Gesamtergebnis ein

  Die Anzahl gewichtet die einzelnen Fehlertypen, alle Fehlertypen haben dasselbe Gewicht

Vorbemerkung 2

BSSE System and Software Engineering

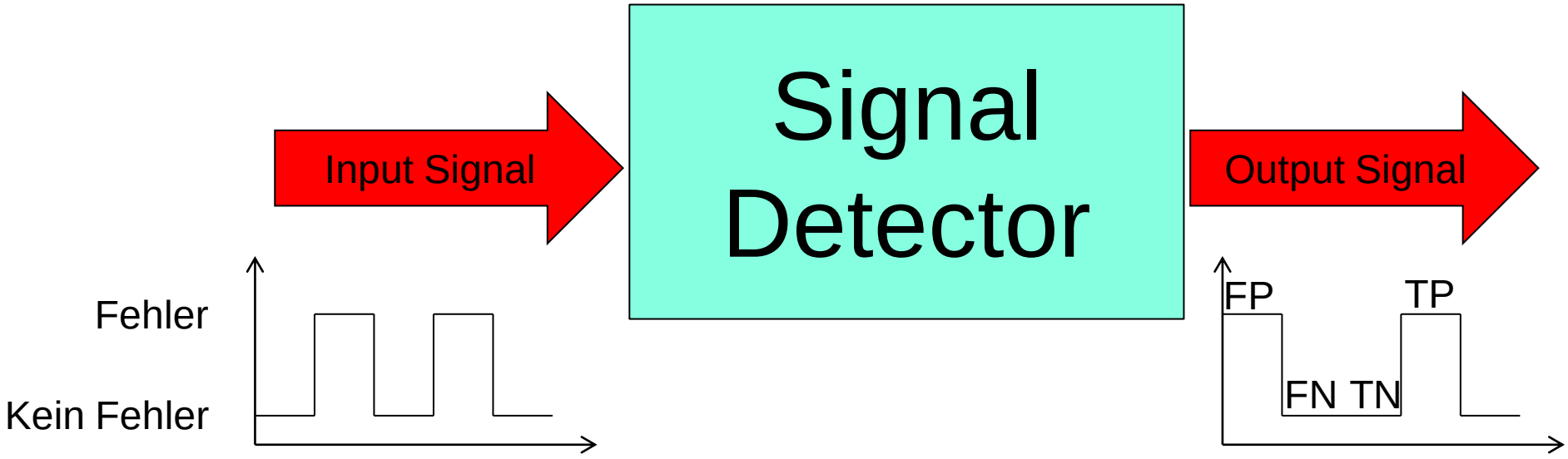
DLR

Bewertung einer Meldung von einem Werkzeug:

Ist der gemeldete Fehler formal auslösbar?

Es geht nicht um die Bewertung der Meldung aus Sicht der Anwendung.

Einstufung von Meldungen



		Output	
		False	True
Input	False	True Negative	False Positive
	True	False Negative	True Positive

Green, D.M., Swets, J.A. (1966) Signal Detection Theory and Psychophysics. New York: Wiley

Defekte, Fehler und Meldungen

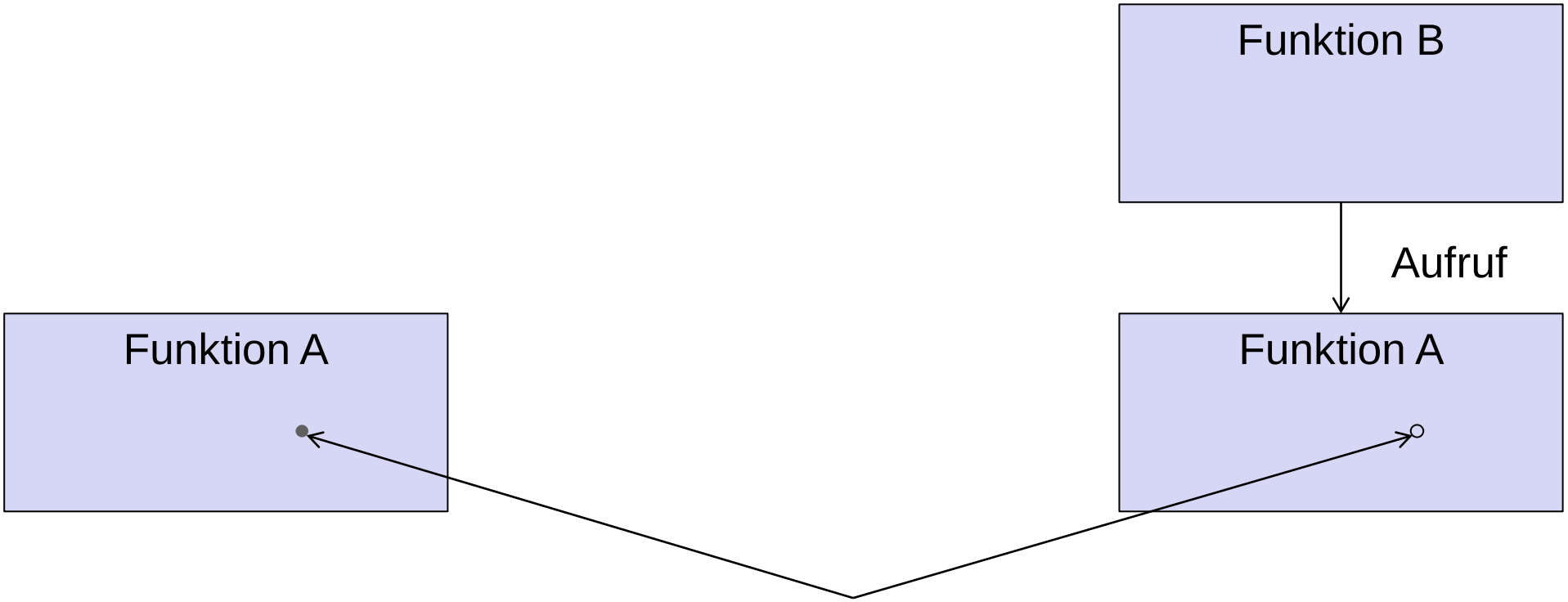
BSSE System and Software Engineering

DLR

Term	Ursache und Wirkung
Fault	Fehler im Code
⇓	⇓
Error	Invalider Systemzustand
⇓	⇓
Failure	Unerwartetes, beobachtbares Verhalten

Ein Defekt bezieht sich auf ein Problem in der Software, entweder bzgl. des externen Verhaltens oder innerer Eigenschaften (wie Wartbarkeit)

Abhängigkeit der Bewertung vom Aufrufkontext



True Positive

False Positive

Aufrufkontext beschränkt die Zustandsmenge

(False Positive)

(True Positive)

(Sonderfälle: Unreachable Code, Invariant Expression, …)

Beispiel Aufrufkontext

```
errorCode_t enableMonitoring( const byte_t * buffer,
        const uint32_t buffer_size, enableMonitoring_t * tc ) {
    uint32_t start = 0;
    if ((buffer == NULL_POINTER) || (tc == NULL_POINTER))
        <error>
     else {
        if ( buffer_size < 1 )  // wrong check!!
           <error>
        else {
           tc->elemNo = buffer[start];
          start += 1;   // invariant expression !!
          upLim = tc->elemNo;
          if (upLim>PARA_MAX)
             upLim=PARA_MAX;
          ii=0;
          while (upLim > ii){
            read32(&buffer[start], &tc->para[ii].elem);
            start += 4;
            ii++;
        }
    }
    return ret;
}
```

Prüfung auf NULL-Pointer führt im Kontext zu Unreachable Code, wenn Funktion niemals mit NULL-Pointer aufgerufen wird.

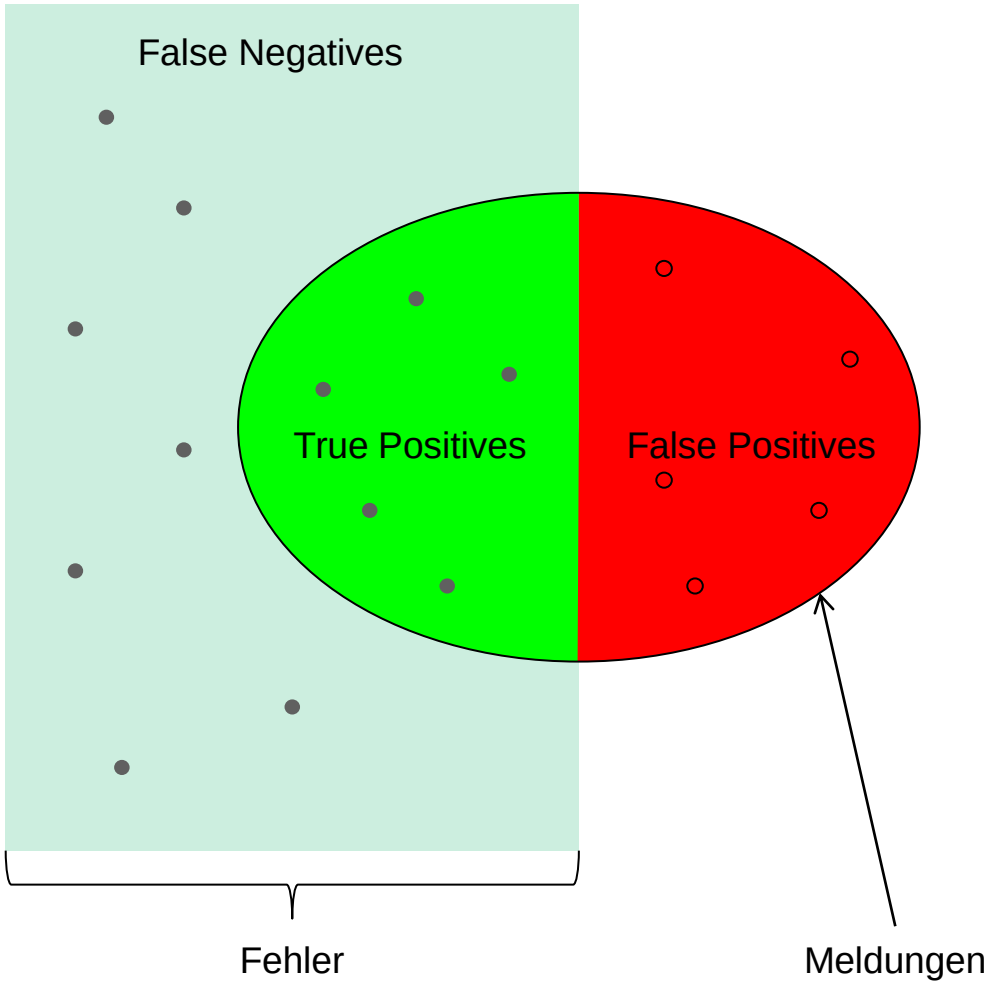
Die betrachteten normierten Defekttypen

BSSE System and Software Engineering

DLR

Defekttyp	Kritikalität
Array Index Out-of-Bounds	Critical
Dereference of Invalid Pointer	Critical
Dereference of NULL-Pointer	Critical
File Access Error	Critical
Invalid function pointer	Critical
Non-terminating Loop	Critical
Passing invalid argument to standard library routine	Critical
(Possible) Recursion	Critical
Resource Leak	Critical
Undefined Result of Arithmetic Operation	Critical
Uninitialized Variable	Critical
Arithmetic Operation on NULL Pointer	Warning
Invariant Condition	Warning
Invariant Expression	Warning
Parameter Type Mismatch in Function Call	Warning
Timeout during execution	Warning
Unnecessary loop construct	Warning
Unreachable Code	Warning
Unused Result	Warning
Multiple return paths	Uncritical

Sensitivität und Genauigkeit



		Output	
		False	True
Input	False	True Negative	False Positive
	True	False Negative	True Positive

$$\text{Genauigkeit} = \frac{\text{TP}}{\text{TP+FP}}$$

Wieviele der Meldungen sind echte Fehler?

100% Genauigkeit ⇒ keine False Positives

$$\text{Sensitivität} = \frac{\text{TP}}{\text{TP+FN}}$$

Wieviele der Fehler wurden gemeldet?

100% Sensitivität ⇒ keine False Negatives

Auswirkungen von False Positives

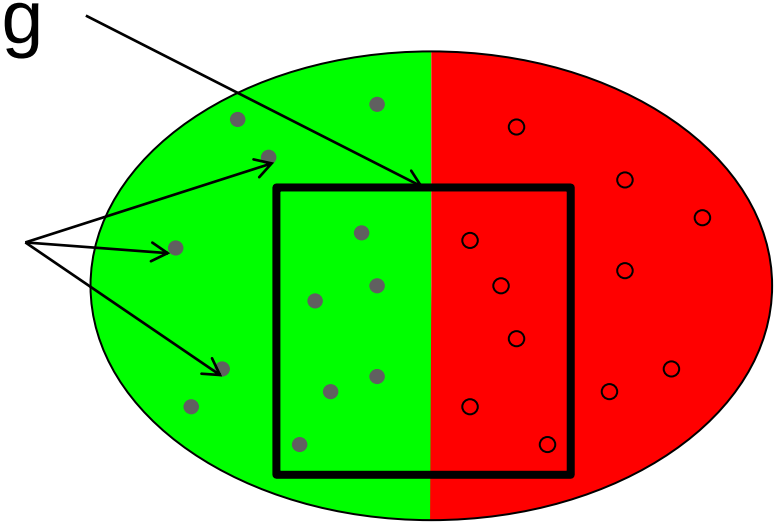
Zu viele Meldungen, zu wenig Zeit

⇒Nutzer analysieren nur einen Auszug
⇒Hersteller
- melden „weiß nicht“-Fälle nicht
- weichen konservative Aufnahmen auf

⇒mehr effektive False Negatives

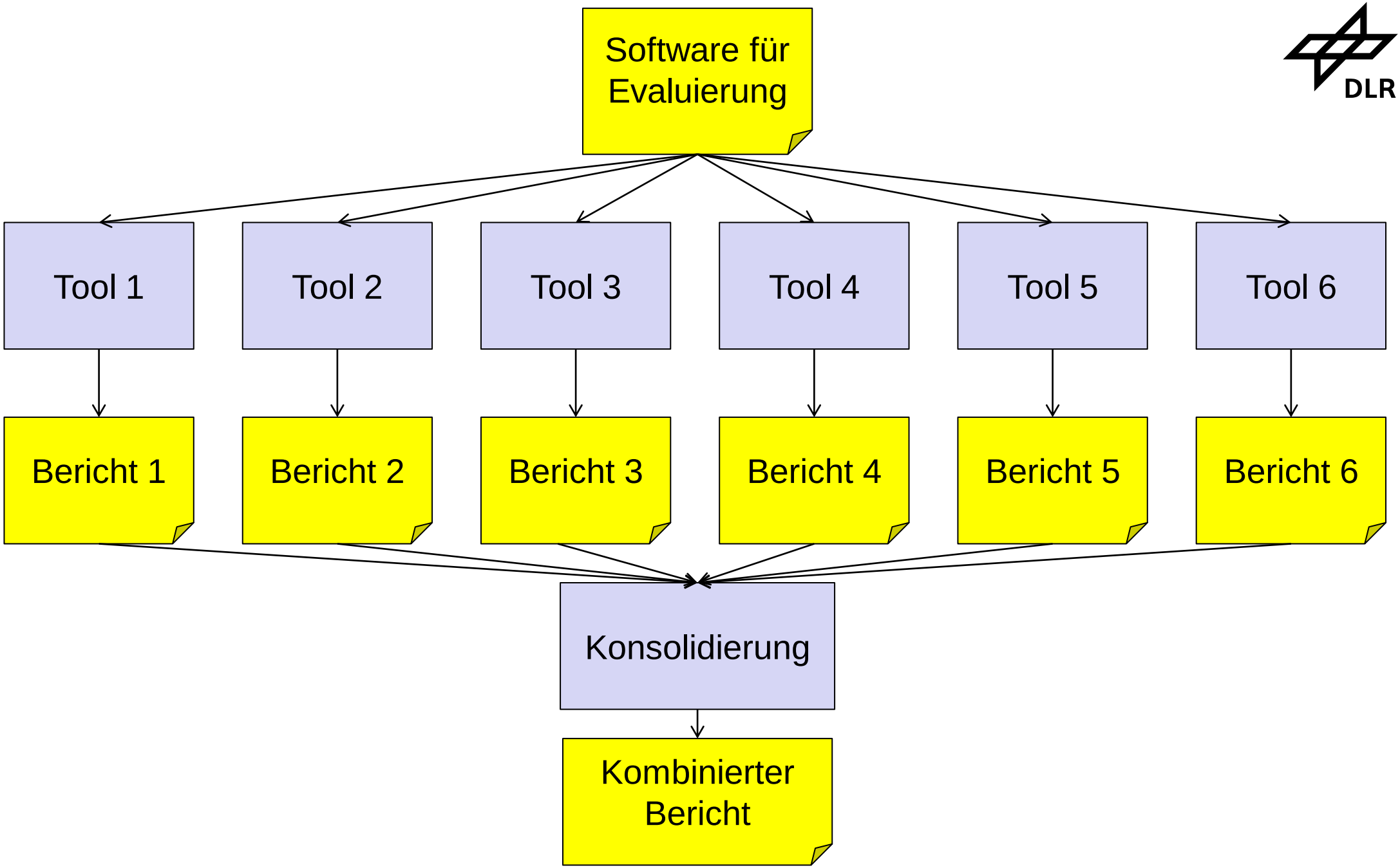
Analysierter Auszug

Effektive False Negatives

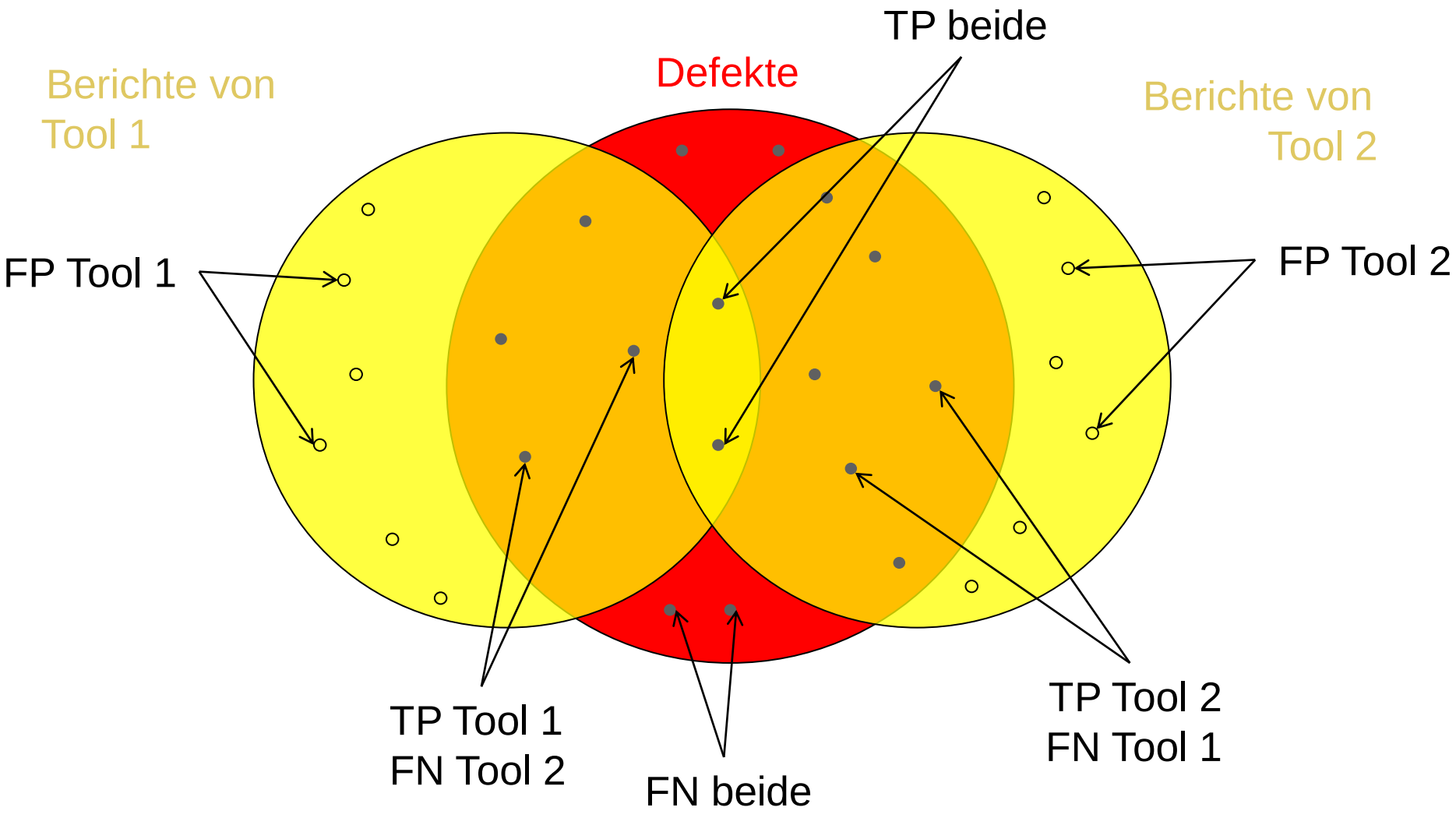


Projekt

Vorgehensweise



Vergleich der Meldungen



Die tatsächliche Menge an Fehlern bleibt unbekannt!

Die Software

Total	
Property	**Quantity**
Size / KLOC	42
Functions, total	610
API Functions	376
c-Files	39
h-files	96

Ausgewertet 6 Werkzeuge, jeweils 4 Kombinationen	
Property	**Quantity**
Size / KLOC	4
Functions, random	31
Functions, by fault distribution	29
Functions, total	60
c-Files	24

Tool	1	2	3	4	5	6
Reports	146	742	1481	4995	2106	9

Defekt-Statistik

BSSE System and Software Engineering

Grouping		Anzahl Meldungen				
		TP+FP	TP		FP	
			mit ctxt	ohne ctxt	mit ctxt	ohne ctxt
Non-Grouped	Alle Defekttypen	500	369	381	131	119
	weighted	439	311	320	129	119
Grouped	Alle Defekttypen	270	195	201	75	69
	weighted	231	159	162	72	69

Methoden der Werkzeuge

Tool	Characteristiken	
	Typ	Methode
1	statisch	abstrakte Interpretation + Meldungsheuristik
2	statisch	abstrakte Interpretation
3	dynamisch	Test, Auto-Stimulation
4	statisch	symbolische Ausführung, Datenflussanalyse
5	statisch	symb. Model-Checking + Datenflussanalyse
6	statisch/ Compiler	Syntax- und Typ-Prüfungen

Tool 6: gcc mit –Wall (ohne Optimierung) zum Vergleich

Lessons Learned

Die Werkzeuge erzeugen viele unzutreffende Meldungen!

Falsch!

Verteilung der True Positives

Defektypen	True Pos		Untermenge "weighted"
	mit ctxt	ohne ctxt	
Array Index Out-of-Bounds	120	126	x
Dereference of Invalid Pointer	31	47	x
Dereference of NULL-Pointer	3	8	x
File Access Error	1	1	x
Invalid function pointer	2	2	x
Non-terminating Loop	1	1	x
Passing invalid argument to standard library routine	1	1	x
(Possible) Recursion	1	1	x
Resource Leak	2	2	x
Undefined Result of Arithmetic Operation	2	2	x
Uninitialized Variable	14	15	x
Arithmetic Operation on NULL Pointer	0	3	
Invariant Condition	16	12	x
Invariant Expression	44	44	
Parameter Type Mismatch in Function Call	2	2	x
Timeout during execution	1	2	x
Unnecessary loop construct	1	1	
Unreachable Code	61	45	x
Unused Result	58	58	x
Multiple return paths	12	12	
Total, 60 functions	**373**	**385**	**FP ~ 25%**
Reports in total, 610 functions: 150 .. 5000			

Die Werkzeuge erzeugen viele unzutreffende Meldungen?

Falsch!

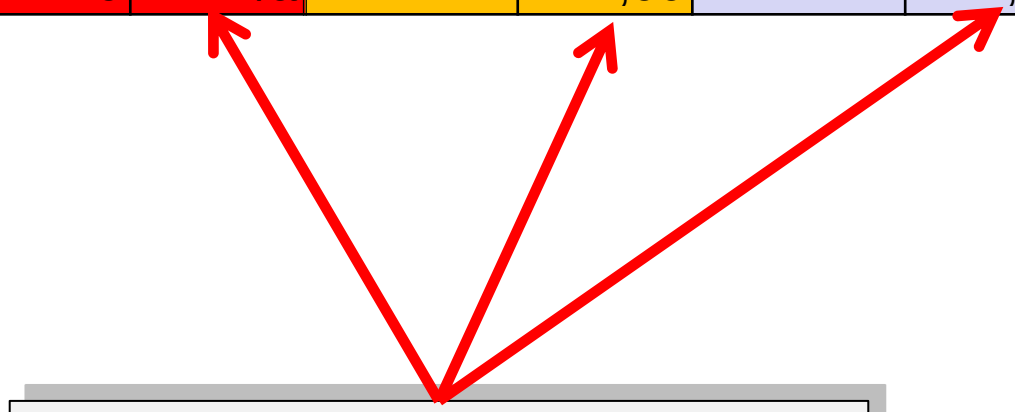
Im Mittel über alle Werkzeuge und Meldungen nur 25%

Anzahl Meldungen

TP+FP	TP		FP	
	mit ctxt	ohne ctxt	mit ctxt	ohne ctxt
500	373	385	131	119

Genauigkeit
Mittel über Fehlertypen

Tool	Criticality Level (with context, not grouped)					
	critical		warning		All (weighted)	
	TP+FP	P	TP+FP	P	TP+FP	P
1	9	0,78	21	1,00	30	0,93
2	138	0,51	27	1,00	165	0,59
3	101	0,78	85	0,48	186	0,65
4	55	0,98	43	0,91	98	0,95
5	100	0,78	71	1,00	171	0,78
6	0	n/a	2	1,00	2	1,00



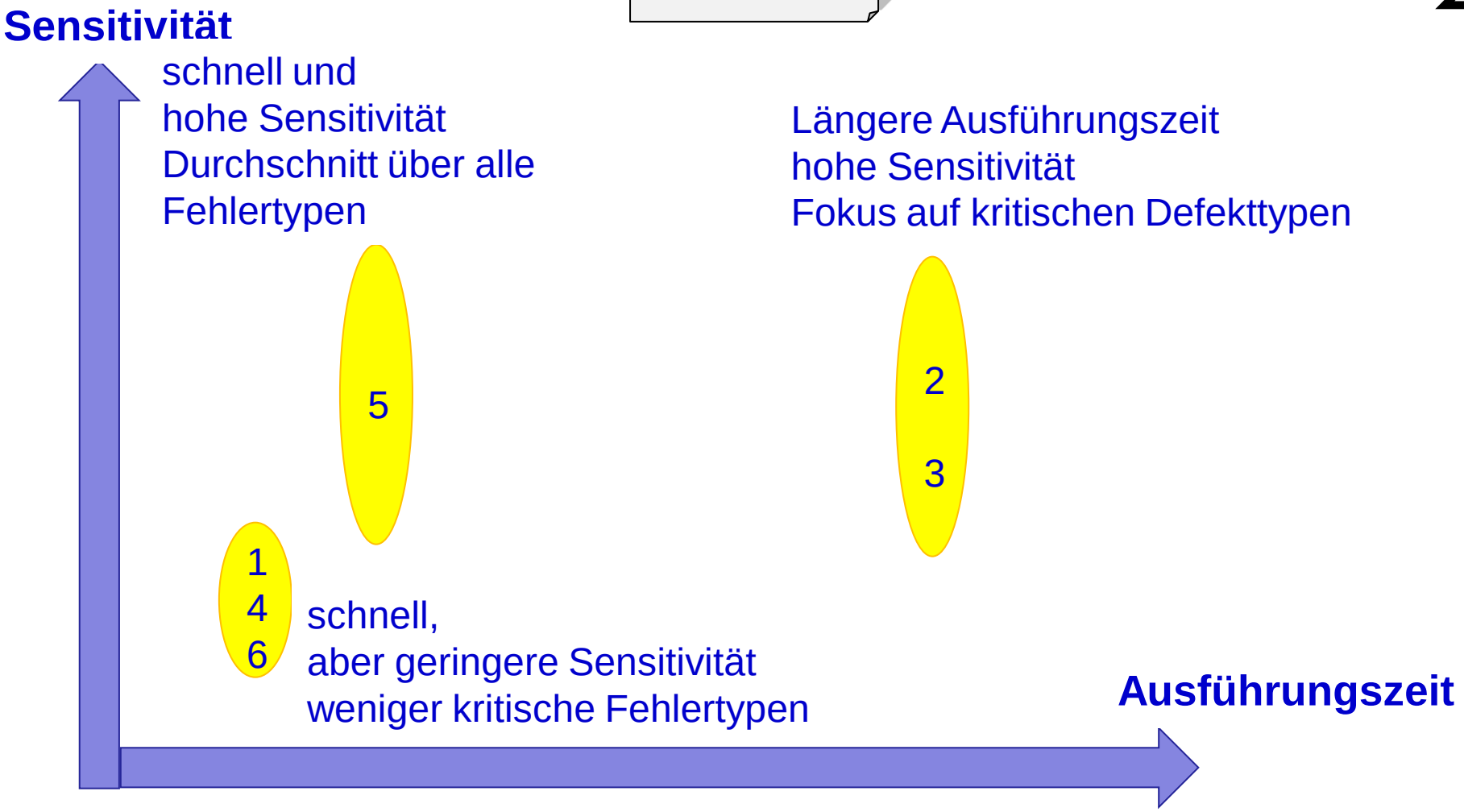
Minimum-Genauigkeit 48-59%, bis zu 100%!

Alle Werkzeuge haben das Ziel, alle Fehler zu finden.

Falsch!

Alle Werkzeuge haben das Ziel, alle Fehler zu finden?

Falsch!



Die Werkzeuge unterscheiden sich nicht.

Falsch!

Ein Werkzeug findet alle relevanten Fehler.

Falsch!

Es reicht, ein Werkzeug einzusetzen.

Falsch!

Die Werkzeuge unterscheiden sich!

BSSE System and Software Engineering

DLR

Defect Type		TP Reported by Tool					
		1	2	3	4	5	6
Array Index Out-of-Bounds		x	x	x	x	x	
Dereference of Invalid Pointer		x	x	x		x	
Dereference of NULL-Pointer			x	x		x	
File Access Error				x			
Invalid function pointer			x				
Non-terminating Loop			x			x	
Passing invalid argument to standard library routine			x				
(Possible) Recursion		x	x			x	
Resource Leak					x		
Undefined Result of Arithmetic Operation		x	x			x	
Uninitialized Variable		x	x		x	x	
Arithmetic Operation on NULL Pointer						x	
Invariant Condition			x	x	x	x	
Invariant Expression						x	
Parameter Type Mismatch in Function Call							x
Timeout during execution				x			
Unnecessary loop construct						x	
Unreachable Code		x	x	x	x	x	
Unused Result		x			x	x	
Multiple return paths						x	
Defect Types Supported by a Tool	**all**	**7**	**11**	**7**	**6**	**14**	**1**
	critical	5	9	4	3	7	0
	warning	2	2	3	3	7	1
	uncritical	0	0	0	0	1	0

Kein Werkzeug findet alle Fehler!

Welche Fehler finden Werkzeuge nicht?

(momentan?)

Dieser Fehler kann (momentan) von keinem Werkzeug gefunden werden

```
tc->elemNo_2 = buf[ind];
ind += 1;
upLim2= tc->elemNo_2;
if (upLim2>LENGTH_87)
  upLim2=LENGTH_87;
idx2=0;
while (upLim2 > idx2){
  read4Byte_86(&buf[ind], &tc>elem[idx2].elem0);
  ind += 4;
  tc->elem[idx2].elemNo_1 = buf[ind];
  ind += 1;
  upLim1= tc->elem[idx2].elemNo_1;
  if (upLim1>LENGTH_87)
    upLim1=LENGTH_87;
  idx1=0;
  while (upLim1 > idx1){

   read4Byte_86(&buf[ind],
       &tc>elem[idx2].elem_1[idx1].elem_1);
    ind += 4;
    indx1++;
  }
 }
 indx2++;
}
```

Wrong

Trailing elements will not be skipped

```
upLim1= tc->elem[idx2].elemNo_1;


idx1=0;
while (upLim1 > idx1){
  if (idx1 < LENGTH_87)
    read4Byte_86(&buf[ind],
    &tc->elem[idx2].elem_1[idx1].elem_1);
    ind += 4;
    indx1++;
  }
 }
 indx2++;
 }
```

Correct

Jeder gemeldete Fehler kann auch beseitigt werden.

Falsch!

Übersicht über die Anzahl der Meldungen

Tool	1	2	3	4	5	6
Reports	146	742	1481	4995	2106	9

anfangs ~ 95.000 Meldungen

hauptsächlich lexikalische Regelverletzungen

Fehler, die von der Implementierung eines Werkzeugs unterstützt werden (sollen), werden immer gemeldet.

Falsch!

- Werkzeuge können die Analyse abbrechen
- Solange Defekte vorliegen, kann nicht damit gerechnet werden, dass alle Defekte bereits gemeldet wurden

Statistiken können täuschen!

BSSE System and Software Engineering

```
#define ESVW_SIZE 10
int ESVWfuncInvArrLg_secondErr() {
  int i,j,arr[ESVW_SIZE],result=0;
  for (i=0;i<ESVW_SIZE;i++)
    arr[i]=0;
  for (i=0;i<ESVW_SIZE;i++)
    for (j=0;j<ESVW_SIZE;j++)
      result+=arr[i+j];
  arr[i]=0;       // is out-of-range here
  arr[0]=result;
  for (i=0;i<ESVW_SIZE;i++)
    printf("arr[%d]=%d \n",i,arr[i]);
  return arr[0];
}
```

Proven	OK	Not	Dead	MayBe	ignored	
94.7%	34	2	0	2	(24)	reported
58.1%	34	2	0	2	24	actual

```
#define ESVW_SIZE 10
int ESVWfuncInvArrLgCorrect(){
  int i,j,arr[2*ESVW_SIZE],result=0;
  for (i=0;i<(2*ESVW_SIZE);i++)
    arr[i]=0;
  for (i=0;i<ESVW_SIZE;i++)
    for (j=0;j<ESVW_SIZE;j++)
      result+=arr[i+j];

  arr[0]=result;
  for (i=0;i<(2*ESVW_SIZE);i++)
    printf("arr[%d]=%d \n",i,arr[i]);
  return arr[0];
}
```

Proven	OK	Not	Dead	MayBe	ignored
100.0%	62	0	0	0	0

Es reicht, ein Werkzeug einfach nur anzuwenden.

Falsch!

- Um Meldungen zu verstehen, muss man wissen, auf welcher Grundlage das Werkzeug arbeitet
- Das gilt auch für Meldungen, die erwartet werden, aber nicht erfolgen

Unerwartetes Shadowing

```
char bufout[128], bufin[128];
void docopy(unsigned char size) {
        memcpy(bufout, bufin, size);
}
```

Fehler: Wert von size nicht geprüft, kann >128 sein.

Toolmeldung:

- Zugriff auf Ziel problematisch
- Zugriff auf Quelle in Ordnung (!)

← Ab hier wird size≤128 angenommen!

Es reicht, ein Werkzeug am Projektende zur Abnahme einzusetzen.

Falsch!

Übersicht über die Anzahl der Meldungen

Tool	1	2	3	4	5	6
Reports	146	742	1481	4995	2106	9

anfangs ~ 95.000 Meldungen

Aber auch tausende Meldungen sind zu viel!

Wenn Regeln nicht von Anfang an beachtet werden,

überrascht es nicht, wenn viele Meldungen am Ende erscheinen

Meldungen sind immer verständlich und nachvollziehbar.

Falsch!

Erinnerung: Unerwartetes Shadowing


```
char bufout[128], bufin[128];
void docopy(unsigned char size) {
        memcpy(bufout, bufin, size);
}
```

Fehler: Wert von size nicht geprüft, kann >128 sein.

Toolmeldung:

- Zugriff auf Ziel problematisch ← Ab hier wird size≤128 angenommen!
- Zugriff auf Quelle in Ordnung (!)

Die Meldungen sind eindeutig hinsichtlich der Identifikation eines tatsächlichen Fehlers.

Falsch!

Abhängigkeit vom Kontext

```
errorCode_t enableMonitoring( const byte_t * buffer,
        const uint32_t buffer_size, enableMonitoring_t * tc ) {
    uint32_t start = 0;
    if ((buffer == NULL_POINTER) || (tc == NULL_POINTER))
        <error>
     else {
        if ( buffer_size < 1 )  // wrong check!!
           <error>
        else {
           tc->elemNo = buffer[start];
          start += 1;   // invariant expression !!
          upLim = tc->elemNo;
          if (upLim>PARA_MAX)
             upLim=PARA_MAX;
          ii=0;
          while (upLim > ii){
            read32(&buffer[start], &tc->para[ii].elem);
            start += 4;
            ii++;
        }
    }
    return ret;
}
```

mit Kontext:

Regelverletzung bei defensiver Programmierung

Meldung zu nicht erreichbarem Code bei Fehlerbehandlung

Meldung	Berücksichtigung von Kontext für buffer / tc	
	mit NULL nicht möglich	ohne NULL möglich
ja	true positive	false positive
nein	true negative	false negative

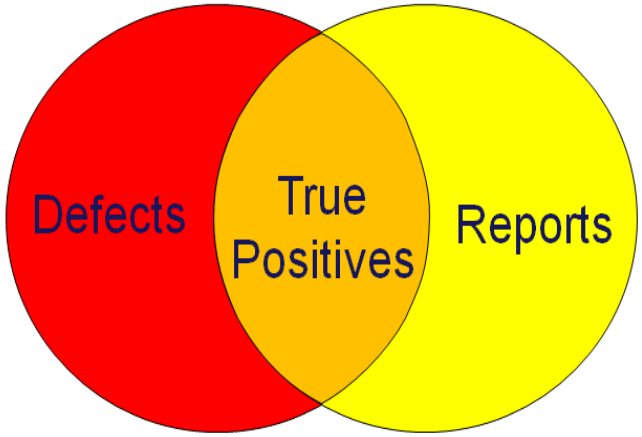
Ergebnisse

Sensitivität und Genauigkeit

Je höher, je mehr Fehler gefunden

Je höher, desto weniger FP

Sensitivity=
TP/Defects



Precision=
TP/Reports

Defekte = TP + FN

Meldungen = TP + FP

Sensitivität and Genauigkeit
Mittel über Fehlertypen

BSSE System and Software Engineering

DLR

Tool	Criticality Level (with context, not grouped)								
	critical			warning			All (weighted)		
	TP+FP	S	P	TP+FP	S	P	TP+FP	S	P
1	9	0,04	0,78	21	0,16	1,00	30	0,09	0,93
2	138	0,40	0,51	27	0,20	1,00	165	0,32	0,59
3	101	0,44	0,78	85	0,31	0,48	186	0,39	0,65
4	55	0,30	0,98	43	0,29	0,91	98	0,30	0,95
5	100	0,44	0,78	71	0,53	1,00	171	0,48	0,78
6	0	0	n/a	2	0,02	1,00	2	0,01	1,00

Sensitivität bei Kombinationen

BSSE System and Software Engineering

DLR

Sensitivity if Tool B added (weighted, with context, not grouped)						
Tool A in Use	**1**	**2**	**3**	**4**	**5**	**6**
1	0,09	0,34	0,44	0,33	0,54	0,10
2	0,34	0,32	0,57	0,52	0,68	0,32
3	0,44	0,57	0,39	0,61	0,77	0,39
4	0,33	0,52	0,61	0,30	0,65	0,31
5	0,54	0,68	0,77	0,65	0,48	0,49
6	0,10	0,32	0,39	0,31	0,49	0,01

Sensitivity if Tool B added (critical, with context, not grouped)						
Tool A in Use	**1**	**2**	**3**	**4**	**5**	**6**
1	0,04	0,41	0,46	0,33	0,46	0,04
2	0,41	0,40	0,69	0,67	0,65	0,40
3	0,46	0,69	0,44	0,71	0,74	0,44
4	0,33	0,67	0,71	0,30	0,55	0,30
5	0,46	0,65	0,74	0,55	0,44	0,44
6	0,04	0,40	0,44	0,30	0,44	0,00

- ein Werkzeug: bis ~ 40 – 50 %
- zwei Werkzeuge: bis ~ 70 – 80 %

Zusammenfassung und Ausblick

Erfahrungen

BSSE System and Software Engineering

DLR

- **Evaluierung**
  - komplexer und aufwändiger als erwartet
  - kein automatischer Abgleich der Meldungen möglich
  - Expertenwissen über Werkzeuge notwendig
- **Unerwartete Erkenntnisse**
  - weniger FP
  - Tools komplementärer
  - Unterschiedliche Ziele der Tools
- **Sensitivität**
  - Selbst das beste Werkzeug findet nur 50%
  - zwei Werkzeuge besser

Ausblick

- mehr Vielfalt bzgl. Werkzeugen und Software
- 2016
  - Projektfortsetzung in Q4
  - zwei weitere Werkzeuge
  - weiteres Softwarepaket (Middleware, C++)
  - drei der bisherigen Werkzeuge auf neues Softwarepaket
- **2017**
  - weiteres Werkzeug
  - drei der bisherigen Werkzeuge
  - weiteres Softwarepaket (komplette Anwendung, C)

Das Projekt wurde duchgeführt für das DLR Raumfahrtmanagement im Namen des Ministeriums für Wirtschaft und Energie BMWi unter Vertrag Fkz 50 PS 1502

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?

Über die Werkzeuge

Tool	1	2	3	4	5	6
			DCRTT		QA•C PRQA	gcc

Sensitivity if Tool B added (weighted, with context, not grouped)						
Tool A in Use	**1**	**2**	**3**	**4**	**5**	**6**
1	0,09	0,34	0,44	0,33	0,54	0,10
2	0,34	0,32	0,57	0,52	0,68	0,32
3	0,44	0,57	0,39	0,61	0,77	0,39
4	0,33	0,52	0,61	0,30	0,65	0,31
5	0,54	0,68	0,77	0,65	0,48	0,49
6	0,10	0,32	0,39	0,31	0,49	0,01

Nennung der Werkzeugnamen entsprechend der Freigabe durch die Hersteller