SAD 2021
Static Analysis Days
Online 5.-6. Mai 2021

Verifysoft TECHNOLOGY

Testen zur Laufzeit und statische Codeanalyse – zwei komplementäre Verfahren

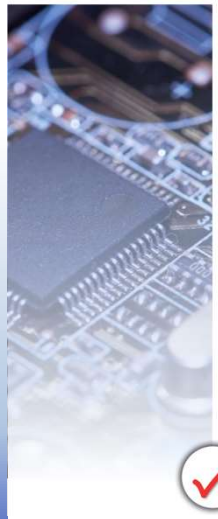
Royd Lüdtke
Verifysoft Technology GmbH
luedtke@verifysoft.com
+49 781 127 8118-0

1

SAD 2021
Static Analysis Days
Online 5.-6. Mai 2021

Agenda

Verifysoft TECHNOLOGY

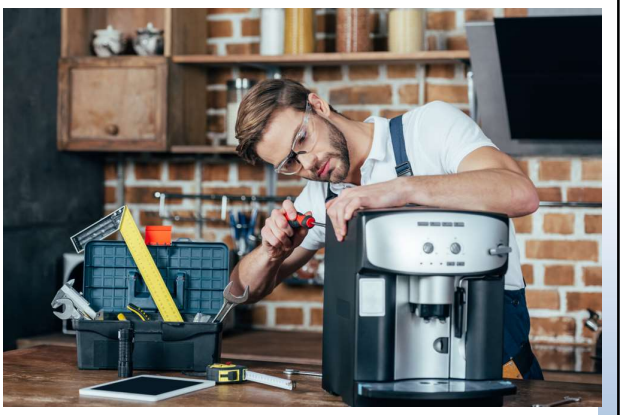


- Fiktives Beispiel: Kaffeeautomat
- Einsatzschwerpunkt für die statische Analyse: Unbestimmtes Verhalten
- Einsatzschwerpunkt für die dynamische Analyse: Funktionale Fehler
- Paradedisziplin der statischen Analyse: Aufdecken von Nebenläufigkeitsfehlern
- Zusammenfassung

2

SAD 2021 Static Analysis Days

verifysoft TECHNOLOGY



Fiktives Beispiel: Fehler in Kaffeeautomat

Foto: www.shutterstock.com

3

SAD 2021 Static Analysis Days

Fiktives Beispiel: Fehler in Kaffeeautomat

verifysoft TECHNOLOGY

```
size_t coffeeCharge(size_t product, size_t cups, size_t hardness) {

  if ((cups < 5) && ((product == 1) || (product == 2))) {
       return hardness * 4;
  }
  else if ((cups < 5) && ((product == 3) || (product == 4))) {
       return hardness * 6;
  }
  else if ((cups < 3) && ((product == 3) || (product == 4))) {
       return hardness * 3;
  }
  else {
       return hardness * 7;
  }
}
```

4

SAD 2021 Static Analysis Days

Fiktives Beispiel: Fehler in Kaffeeautomat

verifysoft TECHNOLOGY

```
size_t getWaterHardness(size_t machine_version) {

  size_t hardness;

  if (machine_version < 7) { //Maschinen ohne Härtesensor
        hardness = 2;
  }
  else if (machine_version > 7) {  //Maschinen mit Härtesensor
        hardness = readSensor();
  }
  return hardness;
}
```


5

SAD 2021 Static Analysis Days

Dynamische Analyse

verifysoft TECHNOLOGY

Test #	Funktion	Eingangsparameter	Erwartetes Ergebnis
1	init()	0	5
2	init()	200000	4
3	init()	a	0
4	init()	-1	3,14
5	getParam()	12, 16, 25	0
6			

Testfälle

Überprüfung zur Laufzeit

Messung der Testabdeckung

Applikation

TER % - MC/DC	TER % - statement	File
Directory: C:\Projects\hcontrol		
79 % - (22/28)	83 % - (20/24)	regulators.c
100 % (8/8)	100 % (4/4)	sensors.c
71 % - (5/7)	93 % - (14/15)	service_functions.c
74 % - (14/19)	75 % - (15/20)	zhome.c
79 % - (49/62)	84 % - (53/63)	DIRECTORY OVERALL
79 % - (49/62)	84 % - (53/63)	OVERALL

Testergebnisse

?
==

Erwartete Ergebnisse

6

SAD 2021 Static Analysis Days

Fiktives Beispiel: Fehler in Kaffeeautomat

verifysoft TECHNOLOGY

Testfall Nr.	machine_version	product	cups	hardness	Ergebnis	erw. Ergebnis
1	6	1	4	2	8	8
2	6	2	4	2	8	8
3	6	1	6	2	14	14
4	6	2	6	2	14	14
5	7	3	4	2	12	12
6	7	4	4	2	12	12
7	8	3	6	1	7	7
8	8	4	6	1	7	7

7

SAD 2021 Static Analysis Days

Fiktives Beispiel: Fehler in Kaffeeautomat

verifysoft TECHNOLOGY

Testfall Nr.	machine_version	product	cups	hardness	Ergebnis	erw. Ergebnis
9	6	3	2	2	12	6
10	6	4	2	2	12	6
11	6	7	4	2	14	14
12	6	3	6	2	14	14

8

SAD 2021 Static Analysis Days

Fiktives Beispiel: Fehler in Kaffeeautomat

verifysoft TECHNOLOGY

CTC++ Coverage Report - Execution Profile

Directory Summary | Files Summary | Functions Summary | Untested Code | |
To files: First | Previous | Next | Last | Index | No Index

Source file: C:\Testwell\Samples\SAD21\SAD21.cpp
Instrumentation mode: multicondition
TER: 82 % (29/35) structural, 97 % (33/34) statement

Hits/True	False	Line	Source
12		30	size_t coffeeCharge(size_t product, size_t cups, size_t hardness) {
2	10	31	if ((cups < 5) && ((product == 1) || (product == 2))) {
1		31	1: (T) && ((T) || (_))
1		31	2: (T) && ((F) || (T))
	5	31	3: (T) && ((F) || (F))
	5	31	4: (F) && ((_) || (_))
2		32	return hardness * 4;
		33	}
4	6	34	else if ((cups < 5) && ((product == 3) || (product == 4))) {
2		34	1: (T) && ((T) || (_))
2		34	2: (T) && ((F) || (T))
	1	34	3: (T) && ((F) || (F))
	5	34	4: (F) && ((_) || (_))
4		35	return hardness * 6;
		36	}
0	6	37	else if ((cups < 3) && ((product == 3) || (product == 4))) {
0		37	1: (T) && ((T) || (_))
0		37	2: (T) && ((F) || (T))
	0	37	3: (T) && ((F) || (F))
	6	37	4: (F) && ((_) || (_))
0		38	return hardness * 3;
		39	}
		40	else {
6		41	return hardness * 7;
		42	}
		43	}
		44	
		45	

9

SAD 2021 Static Analysis Days

Kontrollflussanalyse

verifysoft TECHNOLOGY

Kontrollflussanomalien

- Nicht erreichbare Anweisungen (Dead Code)
- Sprünge aus Schleifen heraus
- Sprünge in Schleifen hinein
- Programmstücke mit mehreren Ausgängen

10

SAD 2021 Static Analysis Days

Fiktives Beispiel: Fehler in Kaffeeautomat

verifysoft TECHNOLOGY

CODESONAR Search code in this analysis for Search | Advanced Search

Home › Kaffeeautomat › Kaffeeautomat analysis 1 › Warning 61977.163756

< Prev (Warning 3 of 3) Next >

Unreachable Computation at SAD21.cpp:32 *No properties have been set.* | edit properties

Jump to warning location ↓ warning details...

Options

coffeeCharge() *C:\Tests\Static_Analysis_Day\SAD21\SAD21\SAD21.cpp*

```
24  size_t coffeeCharge(size_t product, size_t cups, size_t hardness) {
25          if ((cups < 5) && ((product == 1) || (product == 2))) {
26                  return hardness * 4;
27          }
28          else if ((cups < 5) && ((product == 3) || (product == 4))) {
29                  return hardness * 6;
30          }
31          else if ((cups < 3) && ((product == 3) || (product == 4))) {
32                  return hardness * 3;
                    Unreachable Computation
                    The highlighted code will not execute under any circumstances. This may be because of:
                      • A function call that does not return.
                      • A test whose result is always the same: look for a preceding Redundant Condition warning.
                      • A crashing bug. Look for a preceding Null Pointer Dereference or Division By Zero warning.
33          }
34          else {
35                  return hardness * 7;
36          }
37  }
```


11

SAD 2021 Static Analysis Days

Datenflussanalyse

verifysoft TECHNOLOGY

Datenflussanomalien

- Referenzierende Verwendung nicht initialisierter Variablen
- Nichtverwenden eines Variablenwertes

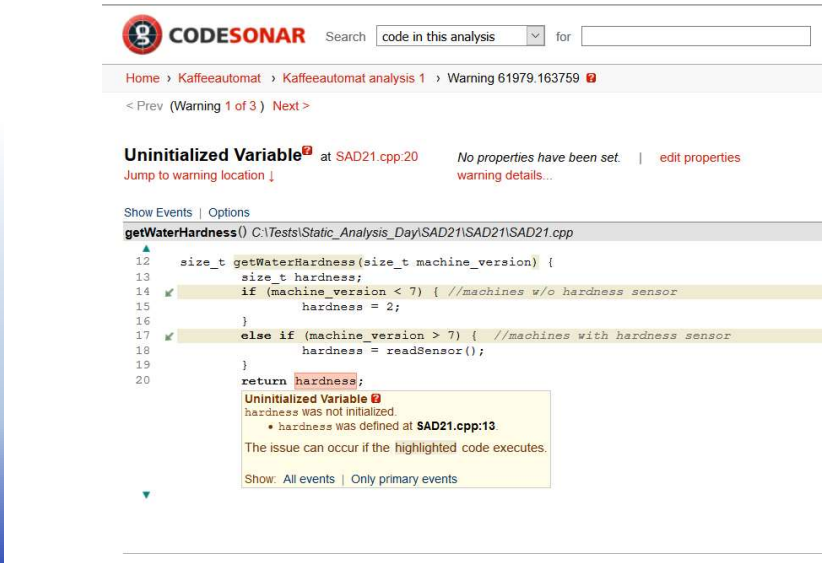
- Auf einem Programmpfad wird ein undefinierter Wert einer Variablen gelesen
- Einer Variablen wird ein Wert zugewiesen. Dieser wird jedoch ungültig, ohne jemals verwendet zu werden
- Eine Variable erhält zum zweiten Mal einen Wert, ohne dass der erste verwendet wurde

12

SAD 2021 Static Analysis Days

Fiktives Beispiel: Fehler in Kaffeeautomat

verifysoft TECHNOLOGY



13

SAD 2021 Static Analysis Days

verifysoft TECHNOLOGY

Einsatzschwerpunkt für die statische Analyse:

Unbestimmtes Verhalten

14

SAD 2021 Static Analysis Days

Memory Leak

verifysoft TECHNOLOGY

```
void simple_leak(void){
  char *p;
  p = (char*)malloc(12);
  if (!p)
       return;
  if (!some_function()){
       free(p);
       return;
  }
  if (!some_other_function())
       return;
  free(p);
  return;
}
```


15

SAD 2021 Static Analysis Days

Buffer Overrun

verifysoft TECHNOLOGY

```
char initBuffer(void){
    char buffer[5];

    buffer[0] = 'A';
    buffer[1] = 'B';
    buffer[2] = 'C';
    buffer[3] = 'D';
    buffer[4] = 'E';
    buffer[5] = 'F';  /* Ueberlauf */

    return buffer[5];

}
```


16

SAD 2021 Static Analysis Days

Double Free

verifysoft TECHNOLOGY

Datei_1.c

```
void double_free(void){

  int *p;

  p = (int*)malloc(sizeof(int));

  if (p){
       afunction(p);
       free(p);
  }
}
```

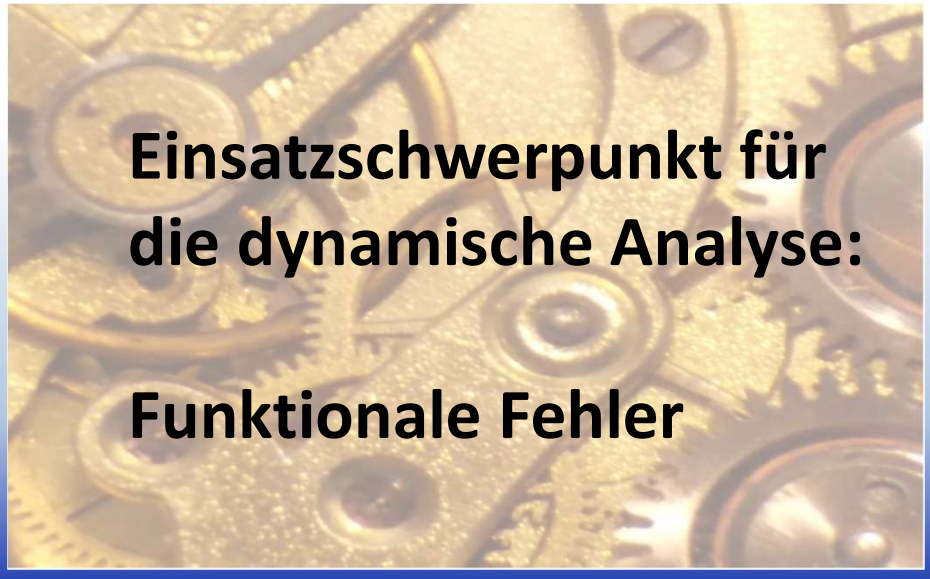
Datei_2.c

```
void afunction(int* p1){
  if (p1)
       free(p1);
}
```


17

SAD 2021 Static Analysis Days

verifysoft TECHNOLOGY



18

SAD 2021 Static Analysis Days

Fehlende Funktionalität

verifysoft TECHNOLOGY

```
void featureSelect(int button) {
   switch (button){
   case 1:
      feature_1();
      break;
   case 2:
      feature_2();
      break;
   case 3:
      feature_3();
      break;
   default:
      return;
   }
}
```

Fehlende Funktionalität

19

SAD 2021 Static Analysis Days

Optimierungsfehler

verifysoft TECHNOLOGY

Originalcode

```
volatile int pressure = 25;

bool checkPressure() {

   if (pressure > 56) {
      openReliefValve();
      return false;
   }
return true;
}
```

Fehlerhafte Optimierung

```
int pressure = 25;

bool checkPressure() {

   return true;
}
```


20

SAD 2021 Static Analysis Days

Endlosschleife

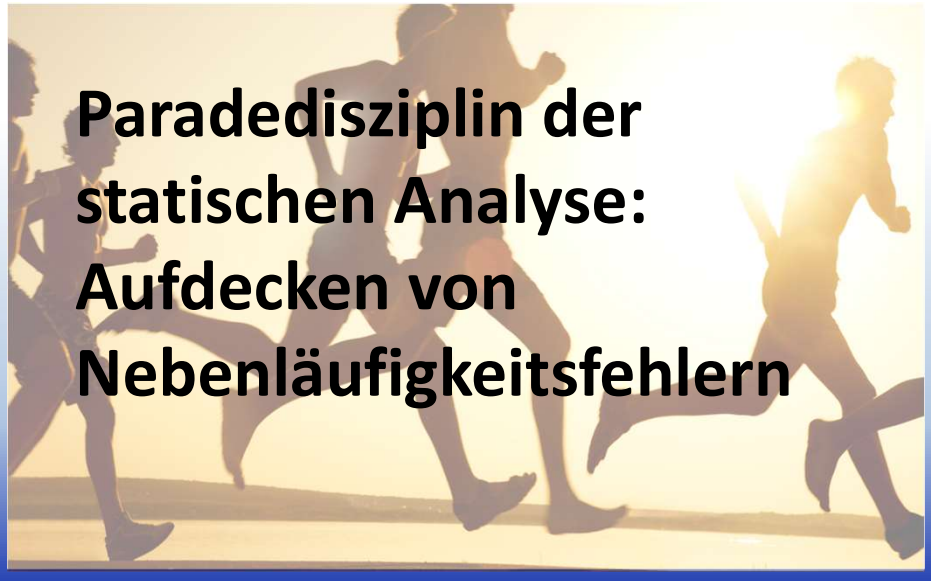
Verifysoft TECHNOLOGY

```
int sum(int number) {

   int i, sum = 0;

   while (i <= number) {

      if (i == 42)
         continue;

      sum = sum + i;

      i++;
   }

   return sum;
}
```


21

SAD 2021 Static Analysis Days

Verifysoft TECHNOLOGY



Paradedisziplin der statischen Analyse: Aufdecken von Nebenläufigkeitsfehlern

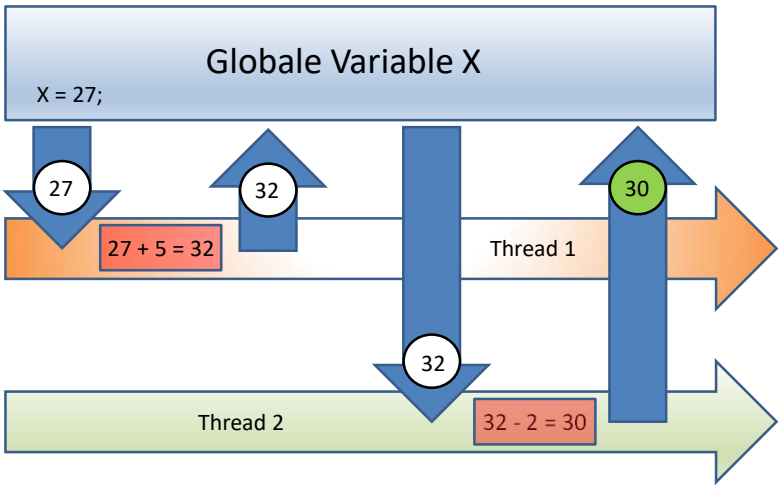
Foto: www.shutterstock.com

22

SAD 2021 Static Analysis Days

Nebenläufigkeitsprobleme
Threads synchronisiert

verifysoft TECHNOLOGY

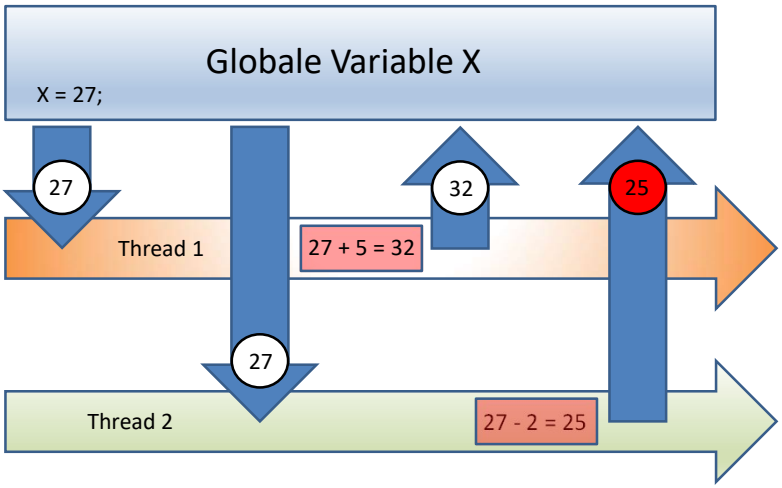


23

SAD 2021 Static Analysis Days

Nebenläufigkeitsprobleme
Threads nicht synchronisiert (Data Race)

verifysoft TECHNOLOGY



24

SAD 2021 Static Analysis Days

Beispiel für Data Race

verifysoft TECHNOLOGY

```
pthread_mutex_t mutex0 = PTHREAD_MUTEX_INITIALIZER;
double sum = 0.;

void *calculate(void *params){
        double d, w;
        unsigned long l;
        unsigned long* part = (unsigned long*) params;
        d = 1.0 / part[2];
        for (l = part[0]; l < part[1]; ++l){
                w = d * (l + 0.5);
/*              pthread_mutex_lock(&mutex0); */
                        sum += 4.0 / (1.0 + w * w);
/*              pthread_mutex_unlock(&mutex0); */
        }
}
```


25

SAD 2021 Static Analysis Days

Beispiel für Data Race

verifysoft TECHNOLOGY

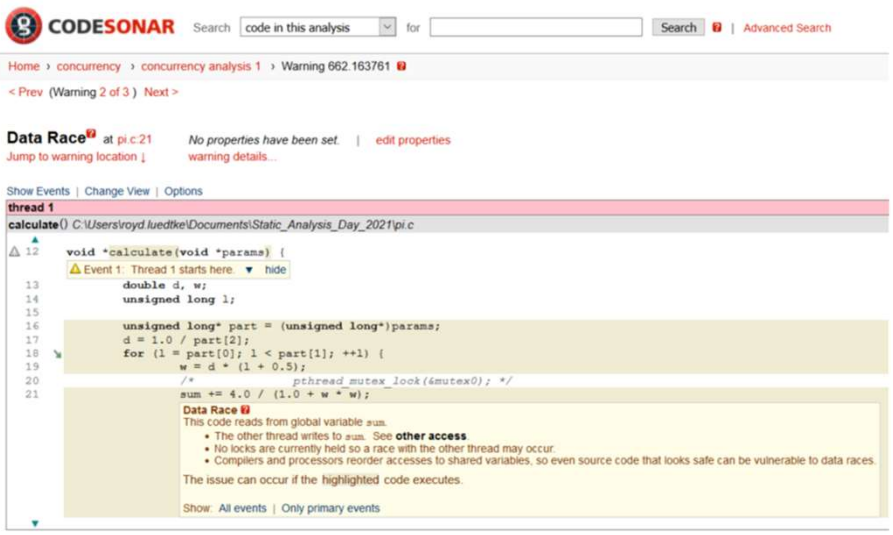
```
int main(void){
        pthread_t pth1, pth2;
        unsigned long segment_info0[3] = {0, 500000, 1000000};
        unsigned long segment_info1[3] = {500001, 1000000, 1000000};
        if (pthread_create(&pth1, NULL, calculate, (void*) &segment_info0) != 0){
                printf("Thread creation failed!\n");
                exit (EXIT_FAILURE);
        }
        if (pthread_create(&pth2, NULL, calculate, (void*) &segment_info1) != 0){
                printf("Thread creation failed!\n");
                exit (EXIT_FAILURE);
        }
        pthread_join(pth1, NULL);
        pthread_join(pth2, NULL);
        printf("PI = %25.20lf\n", sum / segment_info0[2]);
        return 0;
}
```


26

SAD 2021 Static Analysis Days

Beispiel für Data Race

verifysoft TECHNOLOGY

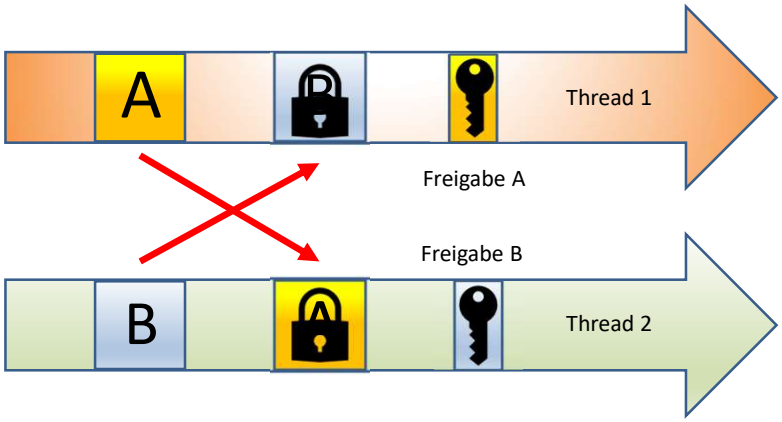


27

SAD 2021 Static Analysis Days

Deadlock

verifysoft TECHNOLOGY



28

SAD 2021 Static Analysis Days

Beispiel für klassischen Deadlock

verifysoft TECHNOLOGY

Funktion durchlaufen von Thread 1

```
void *calculate1(void *params){
        double d, w;
        unsigned long l;
        unsigned long* part = (unsigned long*)params;
        d = 1.0 / part[2];
        for (l = part[0]; l < part[1]; ++l){
                w = d * (l + 0.5);
                pthread_mutex_lock(&mutex0);
                pthread_mutex_lock(&mutex1);
                sum += 4.0 / (1.0 + w * w);
                pthread_mutex_unlock(&mutex1);
                pthread_mutex_unlock(&mutex0);
        }
}
```

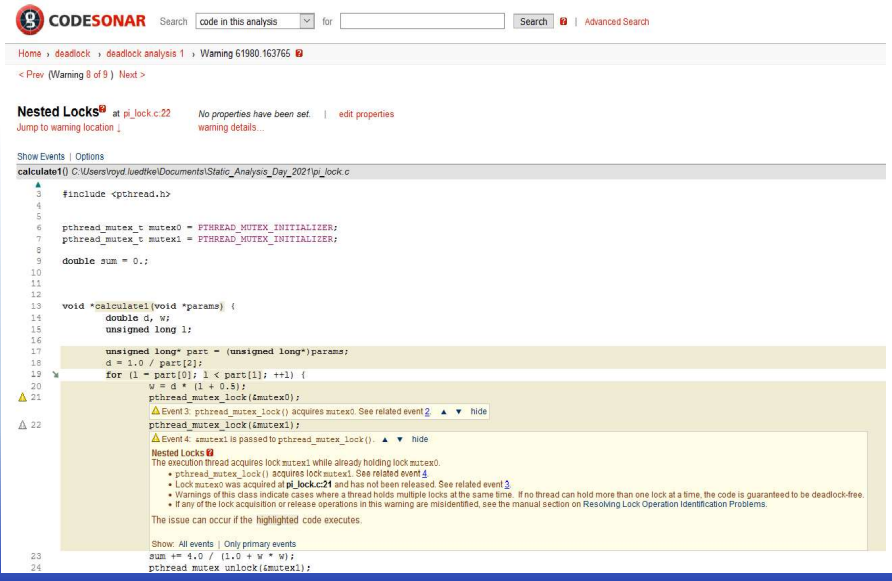
Funktion durchlaufen von Thread 2

```
void *calculate2(void *params){
        double d, w;
        unsigned long l;
        unsigned long* part = (unsigned long*)params;
        d = 1.0 / part[2];
        for (l = part[0]; l < part[1]; ++l){
                w = d * (l + 0.5);
                pthread_mutex_lock(&mutex1);
                pthread_mutex_lock(&mutex0);
                sum += 4.0 / (1.0 + w * w);
                pthread_mutex_unlock(&mutex0);
                pthread_mutex_unlock(&mutex1);
        }
}
```


SAD 2021 Static Analysis Days

Beispiel für klassischen Deadlock

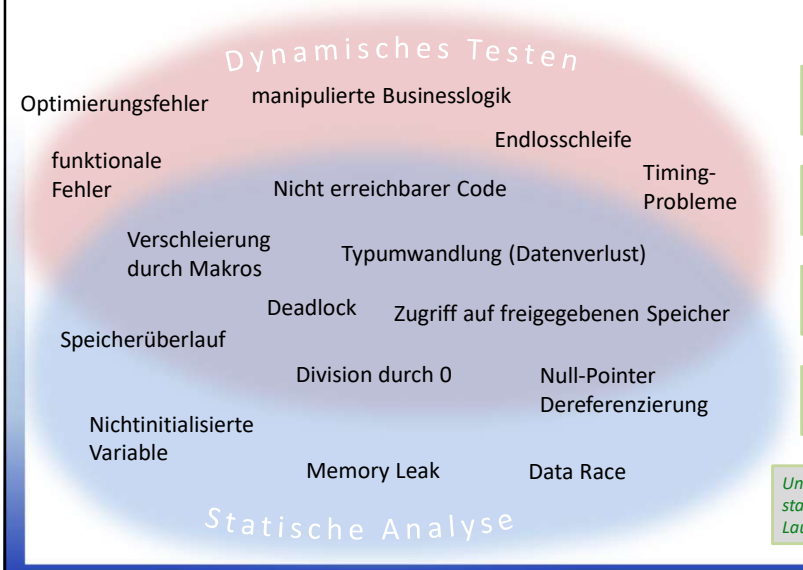
verifysoft TECHNOLOGY



SAD 2021 Static Analysis Days

Zusammenfassung

verifysoft TECHNOLOGY



31

SAD 2021 Static Analysis Days

verifysoft TECHNOLOGY

Das Testen zur Laufzeit triggert Fehler, die sich funktional auswirken. Voraussetzungen für ein sicheres Aufdecken sind das Vorhandensein eines zum jeweiligen Fehler spezifischen Testfalles und ein deterministisches, reproduzierbares Fehlverhalten.

Die statische Quellcodeanalyse beweist ihre Stärke insbesondere beim Auffinden von Fehlern, die zu undefiniertem Verhalten führen.

Nur der gemeinsame Einsatz beider Verfahren ist Voraussetzung für eine hinreichende Softwarequalität.

32

VIELEN DANK !

Royd Lüdtke
Verifysoft Technology GmbH
luedtke@verifysoft.com
+49 781 127 8118-8