Warum White-Box-Test kein Test ist

Andreas Spillner

35. TAV, Ingolstadt

Zur Diskussion gestellt

http://www.swtro.de/fileadmin/swtro/Bilder/Allgemeingueltiges/Diskussion.JPG

Motivation

- „Testgetriebene Entwicklung (Test-Driven Development) bedeutet, dass Tests vor dem Produktivcode geschrieben werden, um so die Softwareentwicklung zu steuern.
  So entsteht Qualitätssoftware mit sehr hoher Testabdeckung, und es
  wird nur das entwickelt, was auch tatsächlich benötigt wird.“

http://www.it-agile.de/agiles-testen.html

Motivation

- „Vorteile von TDD auf einen Blick
- wartbare Qualitätssoftware:
  - kein ungetesteter Code
  - saubere/testbare Architektur durch TDD als Designstrategie
  - keine/wenig Redundanzen durch gnadenloses rechtzeitiges Refaktorisieren“

http://www.it-agile.de/wasisttdd.html

TDD

- „Der Ablauf dieser Programmierung ist zyklisch:
  - Ein Test wird geschrieben, der zunächst fehlschlägt.
  - Genau soviel Produktivcode wird implementiert, dass der Test erfolgreich durchläuft.
  - Test und Produktivcode werden refaktorisiert.“

http://www.it-agile.de/wasisttdd.html

TDD

- „Der Ablauf dieser Program
  zyklisch:
  - Ein Test wird geschrieben
    fehlschlägt.
  - Genau soviel Produktivcode wird implementiert, dass der Test erfolgreich durchläuft.
  - Test und Produktivcode werden refaktorisiert.“

„So entsteht Qualitätssoftware mit sehr hoher Testabdeckung“

100% Codeabdeckung muss sich zwangsläufig ergeben!

!

http://www.it-agile.de/wasisttdd.html

Testen

- Der Prozess, der aus allen Aktivitäten des Lebenszyklus besteht (sowohl statisch als auch dynamisch), die sich mit der Planung, Vorbereitung und Bewertung eines Softwareprodukts und dazugehöriger Arbeitsergebnisse befassen.
  Ziel des Prozesses ist sicherzustellen, dass diese allen festgelegten Anforderungen genügen, dass sie ihren Zweck erfüllen, und etwaige Fehlerzustände zu finden.

ISTQB Glossar

White-Box-Test

- White-Box-Test
  Ein Test, der auf der Analyse der internen Struktur einer Komponente oder eines Systems basiert.
- White-Box-Testentwurfsverfahren
  Ein dokumentiertes Verfahren zur Herleitung und Auswahl von Testfällen, basierend auf der internen Struktur einer Komponente oder eines Systems.

ISTQB Glossar

Abgrenzung

- White-Box-Test
- Beschränkung auf kontrollflussbasierte Ansätze (keine Prüfung von Bedingungen)
- Beschränkung auf Komponententest (TDD-Ebene)

Beispiele

- Maximum von drei Zahlen
- Größter gemeinsamer Teiler
- Dead Code

Maximum von 3 Zahlen finden

Testfälle

```
7 = max(7,5,4)
7 = max(5,7,4)
7 = max(4,5,7)
```

Maximum von 3 Zahlen finden

```
public int max (int x,
       int y,
       int z) {

int max=0;
if (x>z)
   max=x;
if (y>x)
   max=y;
if (z>y)
   max=z;
return max;
}
```

Maximum von 3 Zahlen finden

```
public int max (int x,
                int y,
                int z) {
```

```
max(7,5,4)
int max=0;
if (7>4)
    max=7;
if (5>7)
    max=y;
if (4>5)
    max=z;
return max=7;
}
```

```
max(5,7,4)
int max=0;
if (5>4)
    max=5;
if (7>5)
    max=7;
if (4>7)
    max=z;
return max=7;
}
```

```
max(4,5,7)
int max=0;
if (4>7)
    max=x;
if (5>4)
    max=5;
if (7>5)
    max=7;
return max=7;
}
```

Maximum von 3 Zahlen finden

```
public int max (int x,
                int y,
                int z) {
```

max(7,5,4)

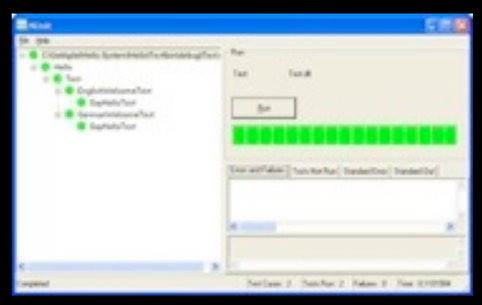
```
int max=0;
if (7>4)
    max=7;
if (5>7)
    max=y;
if (4>5)
    max=z;
return max=7;
}
```

max(5,7,4)

```
int max=0;
if (5>4)
    max=5;
if (7>5)
    max=7;
if (4>7)
    max=z;
return max=7;
}
```

max(4,5,7)

```
int max=0;
if (4>7)
    max=x;
if (5>4)
    max=5;
if (7>5)
    max=7;
return max=7;
}
```



✓ 100% Zweiüberdeckung ✓ Herleitung und Auswahl von Testfällen, basierend auf der internen Struktur einer Komponente oder eines Systems. ?

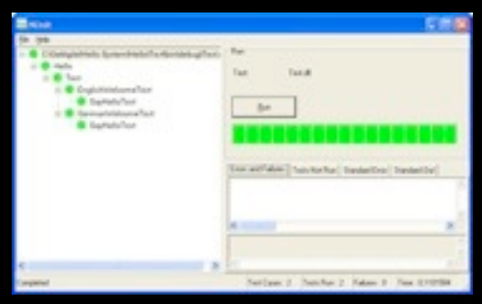
Maximum von 3 Zahlen finden

```
public int max (int x,
                int y,
                int z) {
```

```
max(7,5,4)
int max=0;
if (7>4)
    max=7;
if (5>7)
    max=y;
if (4>5)
    max=z;
return max=7;
}
```

```
max(5,7,4)
int max=0;
if (5>4)
    max=5;
if (7>5)
    max=7;
if (4>7)
    max=z;
return max=7;
}
```

```
max(4,5,7)
int max=0;
if (4>7)
    max=x;
if (5>4)
    max=5;
if (7>5)
    max=7;
return max=7;
}
```



✓ 100% Zweiüberdeckung ✓ Erwartetes Ergebnis aus dem Programmtext ableiten. ?

3 Zahlen finden Maximum von

```
public int max (int x,
       int y,
       int z) {

int max=0;
if (x>z)
    max=x;
if (y>x)
    max=y;
if (z>y)
    max=z;
return max;

}
```

Systematische Testfälle

```
                    7 = max(7,5,4)
x > y > z
x > z > y           7 = max(5,7,4)
y > x > z
y > z > x
z > x > y
z > y > x           7 = max(4,5,7)
x = y > z | y = x > z
x = y < z | y = x < z
x = z > y | z = x > y
x = z < y | z = x < y
y = z > x | z = y > x
y = z < x | z = y < x
x = y = z
```

Maximum von 3 Zahlen finden

```
public int max (int x,
       int y,
       int z) {

int max=0;
if (x>z)
   max=x;
if (y>x)
   max=y;
if (z>y)
   max=z;
return max;

}
```

Systematische Testfälle

```
x > y > z          5 = max(7,4,5)
x > z > y
y > x > z
y > z > x
z > x > y
z > y > x
x = y > z | y = x > z
x = y < z | y = x < z
x = z > y | z = x > y
x = z < y | z = x < y
y = z > x | z = y > x
y = z < x | z = y < x
x = y = z
                   0 = max(7,7,7)
```

3 Zahlen finden

Maximum von

```
public int max (int x,
       int y,
       int z) {

int max=0;
if (x>z)
   max=x;
if (y>x)
   max=y;
if (z>y)
   max=z;
return max;

}
```

13 Testfälle

x	y	z
7	5	4
7	4	5
5	7	4
4	7	5
5	4	7
4	5	7
7	7	5
5	5	7
5	7	5
5	7	7
7	5	5
7	7	7

... allen festgelegten Anforderungen genügen, dass sie ihren Zweck erfüllen, und etwaige Fehlerzustände zu finden.

✓

?

Größter gemeinsamer Teiler

Testfälle

```
2 = gcd(4,6)

2 = gcd(6,4)
```

Größter gemeinsamer Teiler

```
public int gcd(int m, int n) {
    int r;
    if (n > m) {
        r = m;
        m = n;
        n = r;
    }
    r = m % n;
    while (r != 0) {
        m = n;
        n = r;
        r = m % n;
    }
    return n;
}
```

Größter gemeinsamer Teiler

```
2 = gcd(4,6)

public int gcd(int m, int n) {
    int r;
    if (n > m) {
        r = m;
        m = n;
        n = r;
    }
    r = m % n;
    while (r != 0) {
        m = n;
        n = r;
        r = m % n;
    }
    return n;
}
```

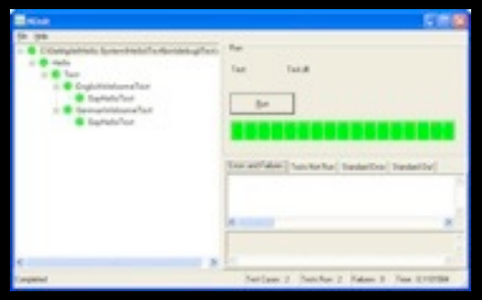
```
2 = gcd(6,4)

public int gcd(int m, int n) {
    int r;
    if (n > m) {
        r = m;
        m = n;
        n = r;
    }
    r = m % n;
    while (r != 0) {
        m = n;
        n = r;
        r = m % n;
    }
    return n;
}
```

Größter gemeinsamer Teiler

```
2 = gcd(4,6)

public int gcd(int m, int n) {
    int r;
    if (n > m) {
        r = m;
        m = n;
        n = r;
    }
    r = m % n;
    while (r != 0) {
        m = n;
        n = r;
        r = m % n;
    }
    return n;
}
```



✓

100%
Zweiüberdeckung

✓

```
2 = gcd(6,4)

public int gcd(int m, int n) {
    int r;
    if (n > m) {
        r = m;
        m = n;
        n = r;
    }
    r = m % n;
    while (r != 0) {
        m = n;
        n = r;
        r = m % n;
    }
    return n;
}
```

... allen festgelegten Anforderungen genügen, dass sie ihren Zweck erfüllen, und etwaige Fehlerzustände zu finden.

?

Größter gemeinsamer Teiler

```
2 = gcd(4,6)

public int gcd(int m, int n) {
    int r;
    if (n > m) {
        r = m;
        m = n;
        n = r;
    }
    r = m % n;
    while (r != 0) {
        m = n;
        n = r;
        r = m % n;
    }
    return n;
}
```

Testfall:
kein gemeinsamer Teiler > 1 fehlt weiterhin
1=(5,4)

!

Herleitung und Auswahl von Testfällen, basierend auf der internen Struktur einer Komponente oder eines Systems.

?

Erwartetes Ergebnis aus dem Programmtext ableiten.

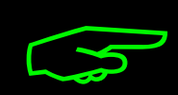
?

Schleifendurchläufe:
- kein Mal 4=(4,4)
- ein Mal 2=(4,6)
- mehrfach 13=(104,169)

?

Dead Code

```
double calculate_price (
double baseprice, double specialprice, double extraprice,
int extras, double discount)
{
double addon_discount; double result;
if (extras≥3) addon_discount=10;
else if (extras≥5) addon_discount=15;
else addon_discount=0;
if (discount>addon_discount) addon_discount=discount;
result  = baseprice/100.0*(100-discount)
        + specialprice
        + extraprice/100.0*(100-addon_discount);
return result;
```



(extras<3)
AND
(extras≥5)

Spillner, A.; Linz, T.: Basiswissen Softwaretest,
5. Auflage, dpunkt Verlag, 2012

Dead Code

```
double calculate_price (
double baseprice, double specialp
in                              )
{
do                                le result,
if                                t=10;
el                                scount
el
if (discount>addon_discount) addo
result  = baseprice/100.0*(100-di
        + specialprice
        + extraprice/100.0*(100-addon_discount);
return result;
```

Symbolische Ausführung
Debugging
...

!

Herleitung und Auswahl von Testfällen, basierend auf der internen Struktur einer Komponente oder eines Systems.

?

Es kann kein Testfall hergeleitet werden, nur der Nachweis, dass es keinen gibt, der das Programmstück zur Ausführung bringt.

!

Spillner, A.; Linz, T.: Basiswissen Softwaretest,
5. Auflage, dpunkt Verlag, 2012

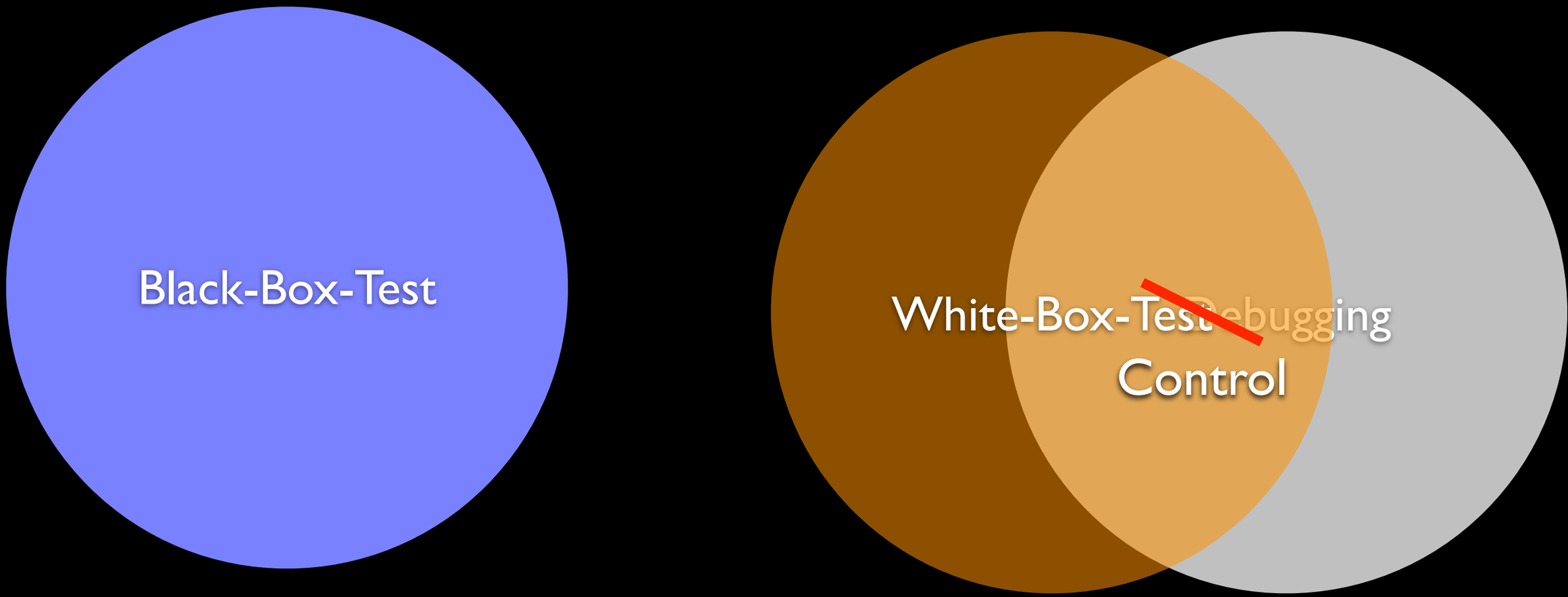
Was stört mich?

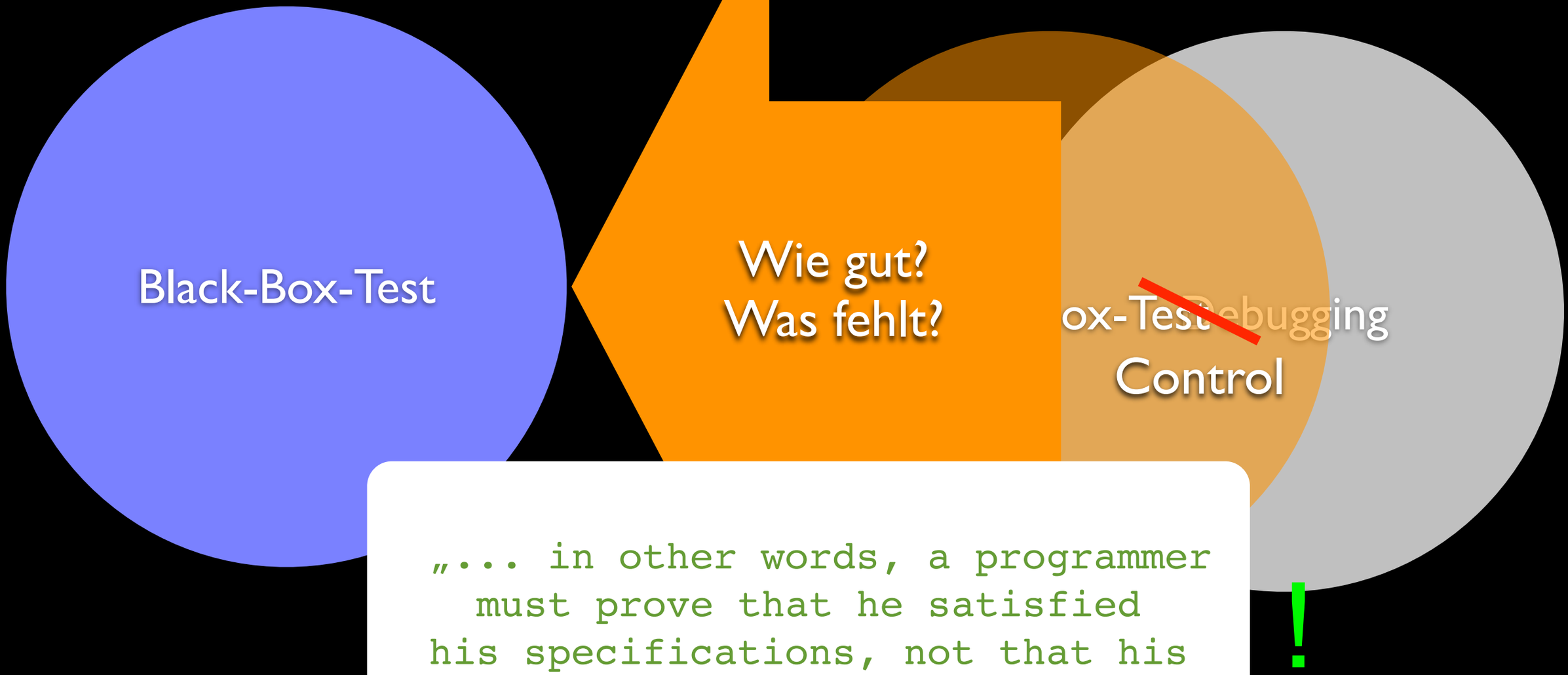
- Codeüberdeckung als
  - Qualitätsnachweis und
  - Testendekriterium, als Testziel
- Black-Box- & White-Box-Verfahren als
  - gleichberechtigte und
  - gleichwertige Verfahren

Black-Box-Test

White-Box-Test

Debugging


Black-Box-Test
White-Box-Test
Debugging
Control



„... in other words, a programmer must prove that he satisfied his specifications, not that his program will perform as coded“

H.D. Benington: Production of Large Computer Programs, Proc. of Symposium On Advanced Computer Programs for Digital Computers, June 1956, Washington, D.C., USA

White-Box-Control

- White-Box-Control
  Kontrolle der durchgeführten Testfälle unter Berücksichtigung der internen Struktur einer Komponente oder eines Systems zur Bewertung der Güte der Testfälle.
- White-Box-Testentwurfsverfahren
  Gibt es nicht =>
  Symbolische Ausführung

Kontakt

- Weiteres gerne auch später ...