

Understanding Software Metrics (ger)

FrOSCon 2010

Manuel Pichler

August 21, 2010



Über mich

- ▶ Manuel Pichler
 - ▶ Total stolzer Papa
 - ▶ Jahrgang 1978
 - ▶ Diplom Informatiker
 - ▶ Gründer Qafoo GmbH - passion for software quality
 - ▶ Training, Consulting & Support zu PHP Software-Qualität
 - ▶ Autor von:
 - ▶ PHP_Depend
 - ▶ PHPMD
 - ▶ phpUnderControl

Outline

Was sind Metriken?

Welche Arten an Metriken gibt es?

Klassische Softwaremetriken

Objektorientierte Softwaremetriken

Zusammenfassung



Softwaremetriken

- ▶ Eine Softwaremetrik ist eine Maßzahl für ein Qualitätsmerkmal von Software
 - ▶ “Funktionen zur Ermittlung von Kennzahlen eines Softwareartefakts” (Wikipedia)

“You cannot control what you cannot measure.” (Tom DeMarco)

- ▶ “...wenn er diese Aussage bereits relativierte, dann kann eine Messung können Auswirkungen durch Änderungen nicht bewertet werden” (A. Fleischer)

Softwaremetriken

- ▶ Eine Softwaremetrik ist eine Maßzahl für ein Qualitätsmerkmal von Software
 - ▶ “Funktionen zur Ermittlung von Kennzahlen eines Softwareartefakts” (Wikipedia)
 - ▶ “You cannot control what you cannot measure.” (Tom DeMarco)
 - ▶ Auch wenn er diese Aussage bereits relativierte
- „...da Messung können Auswirkungen durch Änderungen nicht bewertet werden“ (A. Fleischer)

Softwaremetriken

- ▶ Eine Softwaremetrik ist eine Maßzahl für ein Qualitätsmerkmal von Software
 - ▶ “Funktionen zur Ermittlung von Kennzahlen eines Softwareartefakts” (Wikipedia)
 - ▶ “You cannot control what you cannot measure.” (Tom DeMarco)
 - ▶ Auch wenn er diese Aussage bereits relativierte
 - ▶ “Ohne Messung können Auswirkungen durch Änderungen nicht bewertet werden” (A. Fleischer)

Outline

Was sind Metriken?

Welche Arten an Metriken gibt es?

Klassische Softwaremetriken

Objektorientierte Softwaremetriken

Zusammenfassung



Arten an Metriken

- ▶ Prozessmetrik
- ▶ Aufwandsmetrik
- ▶ Projektlaufzeitmetrik
- ▶ Anwendungsmetrik



Vergesst das aber alles

Wir beschäftigen uns ausschließlich mit Produktmetriken



Outline

Was sind Metriken?

Welche Arten an Metriken gibt es?

Klassische Softwaremetriken

Objektorientierte Softwaremetriken

Zusammenfassung



Umfangsmetriken

- ▶ Summen über Softwareartefakte

- ▶ Lines Of *

- LOC Lines Of Code

- ELOC Executable Lines Of Code

- CLOC Comment Lines Of Code

- NCLOC Non-Comment Lines Of Code

- ▶ Number Of *

- NOC Number Of Classes

- NOM Number Of Methods

- NOP Number Of Packages

Lines Of *, Number Of *

```
1
2 namespace foo\bar;
3
4 abstract class FooBar {
5     abstract function bar();
6 }
7
8 class Foo extends FooBar {
9     /* Does this ... */
10    public function bar() {}
11    /* Does that ... */
12    public function baz() {}
13 }
14
15 class Bar extends Foo {
16     public function foo(Foo $f) {}
17 }
```

► Lines Of *

LOC 14
ELOC 3
CLOC 2
NCLOC 14

► Number Of *

NOC 3
NOM 4
NOP 1

Lines Of *, Number Of *

```
1 namespace foo\bar;  
2  
3 abstract class FooBar {  
4     abstract function bar();  
5 }  
6  
7 class Foo extends FooBar {  
8     /* Does this ... */  
9     public function bar() {}  
10    /* Does that ... */  
11    public function baz() {}  
12 }  
13  
14 class Bar extends Foo {  
15     public function foo(Foo $f) {}  
16 }  
17
```

► Lines Of *

LOC 16

ELOC 3

CLOC 2

NCLOC 14

► Number Of *

NOC 3

NOM 4

NOP 1

Lines Of *, Number Of *

```
1
2 namespace foo\bar;
3
4 abstract class FooBar {
5     abstract function bar();
6 }
7
8 class Foo extends FooBar {
9     /* Does this ... */
10    public function bar() {}
11    /* Does that ... */
12    public function baz() {}
13 }
14
15 class Bar extends Foo {
16    public function foo(Foo $f) {}
17 }
```

► Lines Of *

LOC 16

ELOC 3

CLOC 2

NCLOC 14

► Number Of *

NOC 3

NOM 4

NOP 1

Lines Of *, Number Of *

```
1
2 namespace foo\bar;
3
4 abstract class FooBar {
5     abstract function bar();
6 }
7
8 class Foo extends FooBar {
9     /* Does this ... */
10    public function bar() {}
11    /* Does that ... */
12    public function baz() {}
13 }
14
15 class Bar extends Foo {
16     public function foo(Foo $f) {}
17 }
```

► Lines Of *

LOC 16

ELOC 3

CLOC 2

NCLOC 14

► Number Of *

NOC 3

NOM 4

NOP 1

Lines Of *, Number Of *

```
1
2 namespace foo\bar;
3
4 abstract class FooBar {
5     abstract function bar();
6 }
7
8 class Foo extends FooBar {
9     /* Does this ... */
10    public function bar() {}
11    /* Does that ... */
12    public function baz() {}
13 }
14
15 class Bar extends Foo {
16     public function foo(Foo $f) {}
17 }
```

► Lines Of *

LOC 16

ELOC 3

CLOC 2

NCLOC 14

► Number Of *

NOC 3

NOM 4

NOP 1

Lines Of *, Number Of *

```
1
2 namespace foo\bar;
3
4 abstract class FooBar {
5     abstract function bar();
6 }
7
8 class Foo extends FooBar {
9     /* Does this ... */
10    public function bar() {}
11    /* Does that ... */
12    public function baz() {}
13 }
14
15 class Bar extends Foo {
16     public function foo(Foo $f) {}
17 }
```

► Lines Of *

LOC 16
ELOC 3
CLOC 2
NCLOC 14

► Number Of *

NOC 3
NOM 4
NOP 1

Lines Of *, Number Of *

```
1
2 namespace foo\bar;
3
4 abstract class FooBar {
5     abstract function bar();
6 }
7
8 class Foo extends FooBar {
9     /* Does this ... */
10    public function bar() {}
11    /* Does that ... */
12    public function baz() {}
13 }
14
15 class Bar extends Foo {
16    public function foo(Foo $f) {}
17 }
```

► Lines Of *

LOC 16
ELOC 3
CLOC 2
NCLOC 14

► Number Of *

NOC 3
NOM 4
NOP 1

Lines Of *, Number Of *

```
1
2 namespace foo\bar;
3
4 abstract class FooBar {
5     abstract function bar();
6 }
7
8 class Foo extends FooBar {
9     /* Does this ... */
10    public function bar() {}
11    /* Does that ... */
12    public function baz() {}
13 }
14
15 class Bar extends Foo {
16     public function foo(Foo $f) {}
17 }
```

► Lines Of *

LOC 16
ELOC 3
CLOC 2
NCLOC 14

► Number Of *

NOC 3
NOM 4
NOP 1

Komplexitätsmetriken

- ▶ Maß für Komplexität sind Kontrollstrukturen
 - ▶ if, elseif, for, while, foreach, catch, case, xor, and, or, &&, ||, ?:
- ▶ Cyclomatic Complexity (CCN)
 - ▶ Anzahl der *Verzweigungen*
- ▶ NPath Complexity
 - ▶ Anzahl der *Ausführungspfade*
 - ▶ Berücksichtigt die Struktur von Blöcken

Cyclomatic Complexity

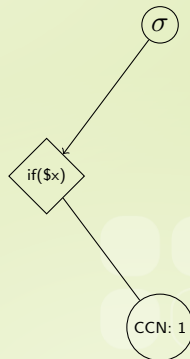
```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```

 σ

CCN: 0

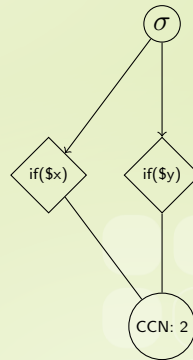
Cyclomatic Complexity

```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```



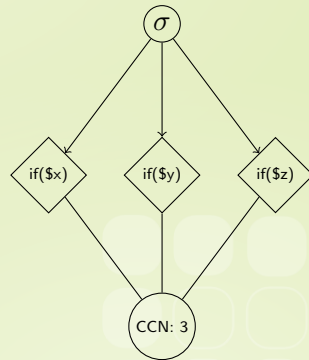
Cyclomatic Complexity

```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```



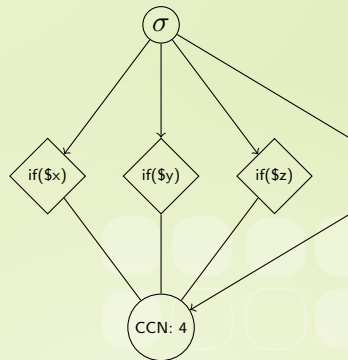
Cyclomatic Complexity

```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```



Cyclomatic Complexity

```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```



NPath Complexity

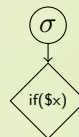
```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```

 σ

NPath: 0

NPath Complexity

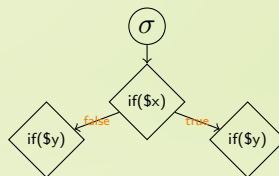
```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```



NPPath: 0

NPath Complexity

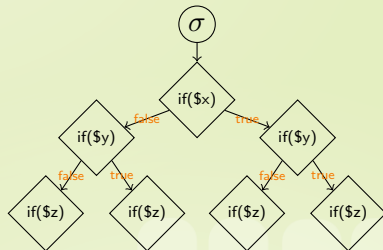
```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```



NPath: 0

NPath Complexity

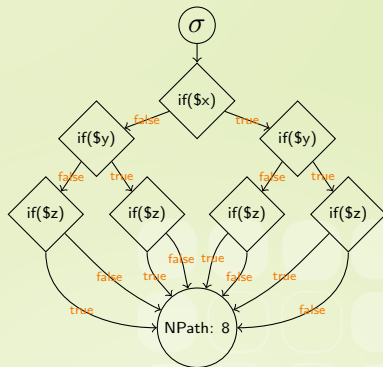
```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```



NPath: 0

NPath Complexity

```
1 <?php
2 class Foo {
3     public function foo() {
4         if ($x) { }
5         if ($y) { }
6         if ($z) { }
7         return $x;
8     }
9 }
```



Grenzwerte

- ▶ Zahlen allein sagen noch nichts aus
 - ▶ Oder was bedeuten die Werte 4 und 8?
- ▶ Für eine Beurteilung benötigt man Grenzwerte
 - ▶ Cyclomatic Complexity
 - ▶ 1-4: low, 5-7: medium, 8-10: high, 11+: hell
 - ▶ NPath Complexity
 - ▶ 200: critical mass
- ▶ Grenzwerte sind immer Ermessenssache

Metriken kombinieren

Die Kombination von Metriken erlaubt einen sehr tiefen Einblick in ein System.

- ▶ LOC: 300; CCN: 42; NOC: 5; NOM: 15
 - ▶ $CCN / LOC = 0,14$
 - ▶ Jede sechste Zeile eine Kontrollstruktur

$$NOC = 5$$

→ prozedural oder große Klassen

$$NOM = 20$$

→ Große Methoden / Funktionen oder prozedural

$$CCN / NOM = 2,8$$

→ Übermäßig komplexe Methoden / Funktionen

Metriken kombinieren

Die Kombination von Metriken erlaubt einen sehr tiefen Einblick in ein System.

- ▶ LOC: 300; CCN: 42; NOC: 5; NOM: 15
 - ▶ $CCN / LOC = 0,14$
 - ▶ Jede sechste Zeile eine Kontrollstruktur
 - ▶ $LOC / NOC = 60$
 - ▶ Primär prozedural oder große Klassen

$NOM = 20$

→ Große Methoden / Funktionen oder prozedural

$CCN / NOM = 2,1$

→ Übermäßig komplexe Methoden / Funktionen

Metriken kombinieren

Die Kombination von Metriken erlaubt einen sehr tiefen Einblick in ein System.

- ▶ LOC: 300; CCN: 42; NOC: 5; NOM: 15
 - ▶ $CCN / LOC = 0,14$
 - ▶ Jede sechste Zeile eine Kontrollstruktur
 - ▶ $LOC / NOC = 60$
 - ▶ Primär prozedural oder große Klassen
 - ▶ $LOC / NOM = 20$
 - ▶ Große Methoden / Funktionen oder prozedural
 - ▶ $CCN / NOM = 2,8$
 - ▶ Übermäßig komplexe Methoden / Funktionen

Metriken kombinieren

Die Kombination von Metriken erlaubt einen sehr tiefen Einblick in ein System.

- ▶ LOC: 300; CCN: 42; NOC: 5; NOM: 15
 - ▶ $CCN / LOC = 0,14$
 - ▶ Jede sechste Zeile eine Kontrollstruktur
 - ▶ $LOC / NOC = 60$
 - ▶ Primär prozedural oder große Klassen
 - ▶ $LOC / NOM = 20$
 - ▶ Große Methoden / Funktionen oder prozedural
 - ▶ $CCN / NOM = 2,8$
 - ▶ Übermäßig komplexe Methoden / Funktionen

Outline

Was sind Metriken?

Welche Arten an Metriken gibt es?

Klassische Softwaremetriken

Objektorientierte Softwaremetriken

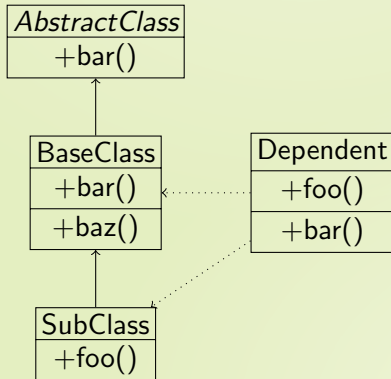
Zusammenfassung



Chidamber & Kemerer

- ▶ Weighted Methods per Class (WMC)
 - ▶ Summe der Komplexität aller Methoden
 - ▶ Grenzwert 20 - 50
- ▶ Number Of Children (NOC)
 - ▶ Anzahl der direkten Ableitungen
 - ▶ Falsch gewählte Abstraktion
- ▶ Depth of Inheritance Tree (DIT)
 - ▶ Vererbungsstrukturen erhöhen die Komplexität
 - ▶ Grenzwert ≤ 5

OO-Metriken



► AbstractClass

WMC 2

DIT 1

► BaseClass

WMC 2

DIT 1

► SubClass

WMC 1

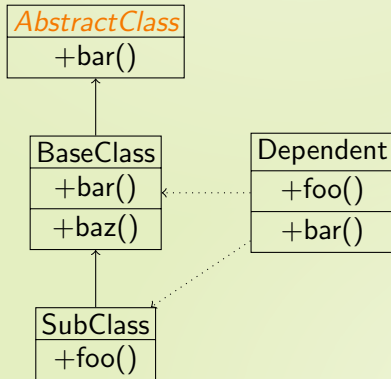
DIT 2

► Dependent

WMC 2

DIT 0

OO-Metriken



► AbstractClass

WMC 0

DIT 0

► BaseClass

WMC 2

DIT 1

► SubClass

WMC 1

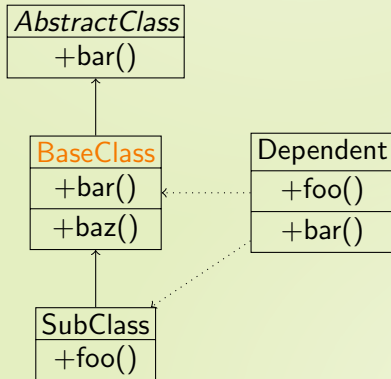
DIT 2

► Dependent

WMC 2

DIT 0

OO-Metriken



► *AbstractClass*

WMC 0

DIT 0

► *BaseClass*

WMC 2

DIT 1

► *SubClass*

WMC 1

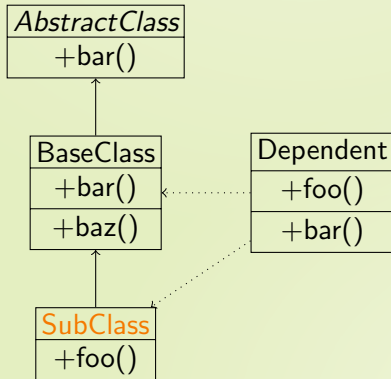
DIT 2

► *Dependent*

WMC 2

DIT 0

OO-Metriken



► AbstractClass

WMC 0

DIT 0

► BaseClass

WMC 2

DIT 1

► SubClass

WMC 1

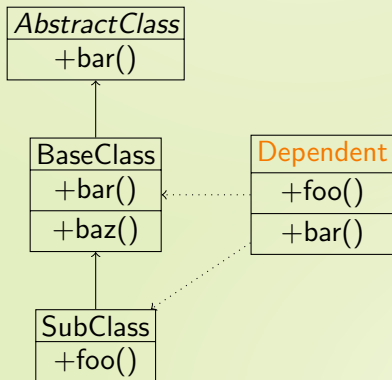
DIT 2

► Dependent

WMC 2

DIT 0

OO-Metriken



► *AbstractClass*

WMC 0

DIT 0

► BaseClass

WMC 2

DIT 1

► SubClass

WMC 1

DIT 2

► **Dependent**

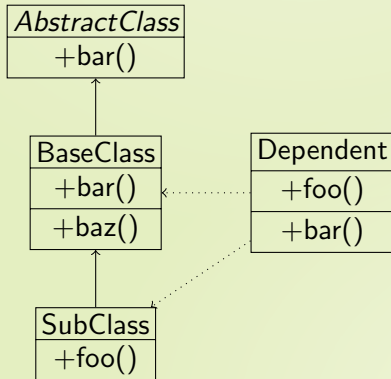
WMC 2

DIT 0

Afferent- & Efferent-Coupling

- ▶ Afferent Coupling (C_a)
 - ▶ Eingehende Abhängigkeiten anderer Komponenten
 - ▶ Großer Einfluss auf die Stabilität des Systems
- ▶ Efferent Coupling (C_e)
 - ▶ Ausgehende Abhängigkeiten:
 - ▶ Vererbung, Parameter, Exceptions, Allokationen
 - ▶ Abhängigkeit von Stabilität anderer Komponenten

OO-Metriken



► *AbstractClass*

Ce 0
Ca 0

► *BaseClass*

Ce 1
Ca 2

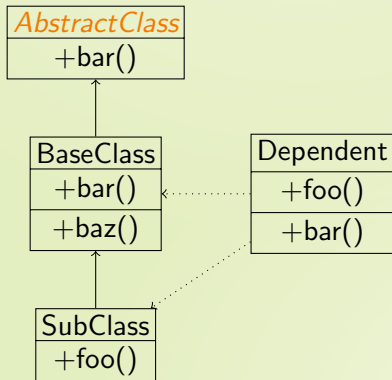
► *SubClass*

Ce 1
Ca 1

► *Dependent*

Ce 2
Ca 0

OO-Metriken



► AbstractClass

Ce 0

Ca 1

► BaseClass

Ce 1

Ca 2

► SubClass

Ce 1

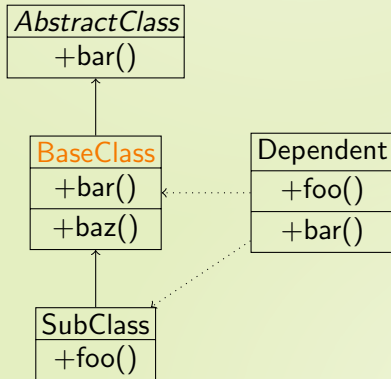
Ca 1

► Dependent

Ce 2

Ca 0

OO-Metriken



► AbstractClass

Ce 0

Ca 1

► BaseClass

Ce 1

Ca 2

► SubClass

Ce 1

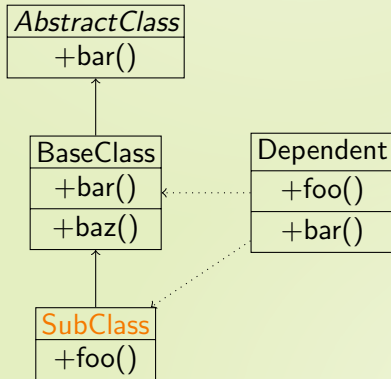
Ca 1

► Dependent

Ce 2

Ca 0

OO-Metriken



► AbstractClass

Ce 0

Ca 1

► BaseClass

Ce 1

Ca 2

► SubClass

Ce 1

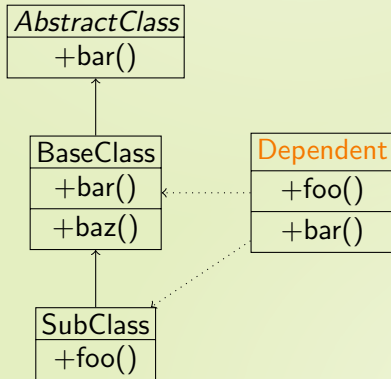
Ca 1

► Dependent

Ce 2

Ca 0

OO-Metriken



► AbstractClass

Ce 0

Ca 1

► BaseClass

Ce 1

Ca 2

► SubClass

Ce 1

Ca 1

► Dependent

Ce 2

Ca 0

Abstraction, Instability & Distance

- ▶ Modell für eine ausgewogene Architektur
 - ▶ Basiert auf Ce, Ca, abstrakten(AC) und konkreten(CC) Softwareartefakten
 - ▶ Abstraction

$$A = AC / (AC + CC)$$

- ▶ Instability

$$I = Ce / (Ce + Ca)$$

- ▶ Distance

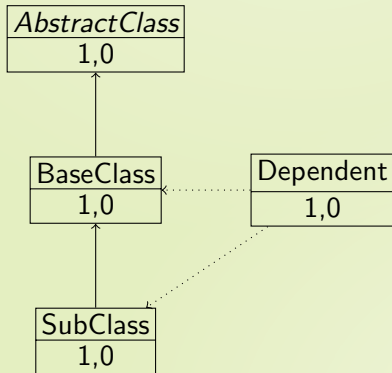
$$D = |A + I| - 1$$

CodeRank

- ▶ Basiert auf dem Google PageRank
- ▶ Software wird auf einen Graphen abgebildet
 - ▶ Knoten (π) je Softwareartefakt
 - ▶ Pakete, Klassen, Methoden
 - ▶ Kanten (ρ) für jede Beziehung
 - ▶ Vererbung, Aufrufe, Parameter, Exceptions
- ▶ Für den CodeRank gilt:

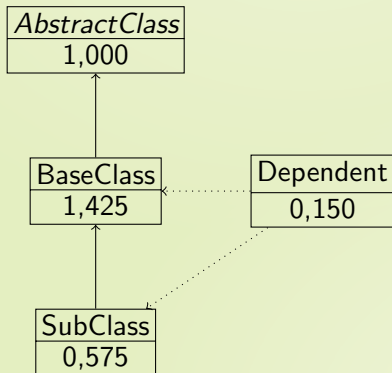
$$CR(\pi_i) = \sum_r r((1 - d) + d \sum_r r(CR(\pi_r)/\rho_r))$$

CodeRank



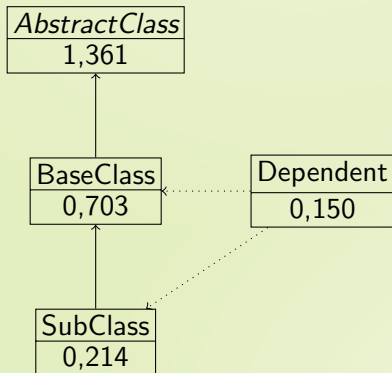
► Iteration: 0

CodeRank



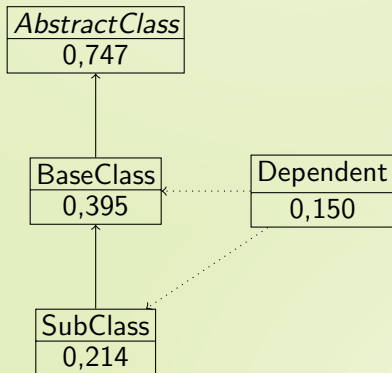
► Iteration: 1

CodeRank



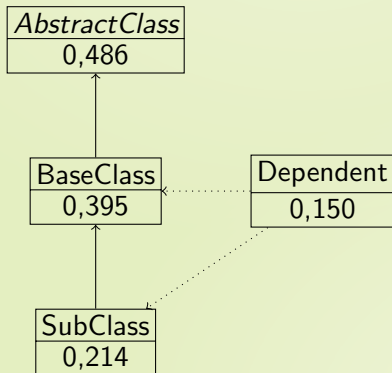
► Iteration: 2

CodeRank



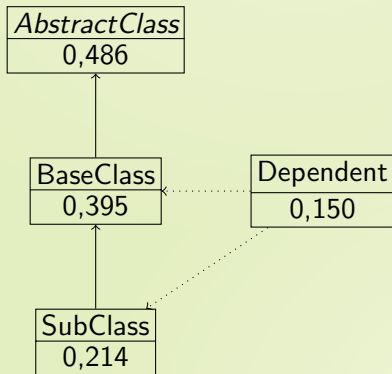
► Iteration: 3

CodeRank



► Iteration: 4

CodeRank



► Iteration: 5

CodeRank

- ▶ Mit dem CodeRank können auch indirekte Abhängigkeiten bewertet werden.
- ▶ Logik mit Einfluss auf das gesamte System kann schnell gefunden werden
- ▶ Äquivalent der Reverse CodeRank
 - ▶ Kann mit dem selben Algorithmus berechnet werden
 - ▶ Gibt Aufschluss über stark abhängige Komponenten

Outline

Was sind Metriken?

Welche Arten an Metriken gibt es?

Klassische Softwaremetriken

Objektorientierte Softwaremetriken

Zusammenfassung



Metriken sind ...

- ▶ ... keine Magie, sondern einfache Maßzahlen
- ▶ ... nutzlos, ohne Grenz- oder Referenzwerte
- ▶ ... skalierbar, wachsen mit der Projektgröße
- ▶ ... automatisierbar und reproduzierbar
- ▶ ... objektiv, da Software gestützt
- ▶ ... interpretierbar, abhängig vom Betrachter

Vielen Dank für das Zuhören

- ▶ Kontakt:
 - ▶ <http://manuel-pichler.de> / <http://qafoo.com>
 - ▶ @manuelp / @qafoo
- ▶ Mehr über uns
 - ▶ <http://qafoo.com>

