Software-Metriken: Purismus vs. Pragmatismus

Web DevCon

Manuel Pichler

17.10.2011

Über mich

- Diplominformatiker

Über mich

- Diplominformatiker
- Mehr als 10 Jahre Erfahrung im Web-Umfeld

Über mich

- Diplominformatiker
- Mehr als 10 Jahre Erfahrung im Web-Umfeld
- Autor hinter einer Reihe von QA-Tools

Über mich

- Diplominformatiker
- Mehr als 10 Jahre Erfahrung im Web-Umfeld
- Autor hinter einer Reihe von QA-Tools

Mitgründer von

We help people to create high quality PHP applications.

`http://qafoo.com`

Outline

Softwaremetriken

- Eine Softwaremetrik ist eine Maßzahl für ein bestimmtes Qualitätsmerkmal von Software

Softwaremetriken

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

Softwaremetriken

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

"You cannot control what you cannot measure."
(Tom DeMarco)

Wer macht hier kein PHP?

PHP to whatever translation table

- `$variable`
- `$this->method()`
- `foreach ( $l as $s )`
- `array(array(1, 2))`
- `public function ...`
- `namespace foo\bar;`
- `<?php`

- `variable`
- `method()` **OR** `this.method()`
- `for ( String s :  l )` `for ( k in l ) s = l[k]`
- `new int[][] {{1, 2}}` `[[1, 2]]`
- `public [void|int|..]  ...`
- `package foo.bar;` `namespace foo.bar {`
- `IGNORE`

Ist aber auch egal...

...

die meisten **Konzepte**

wurden bereits vor

Dekaden entwickelt

Outline

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 *
- Number Of *

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

- Number Of *

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

- Number Of *

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

- Number Of *

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 *

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

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

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

Wichtig

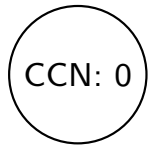
Cyclomatic Complexity

Wichtig

CCN == Anzahl der Verzweigungen

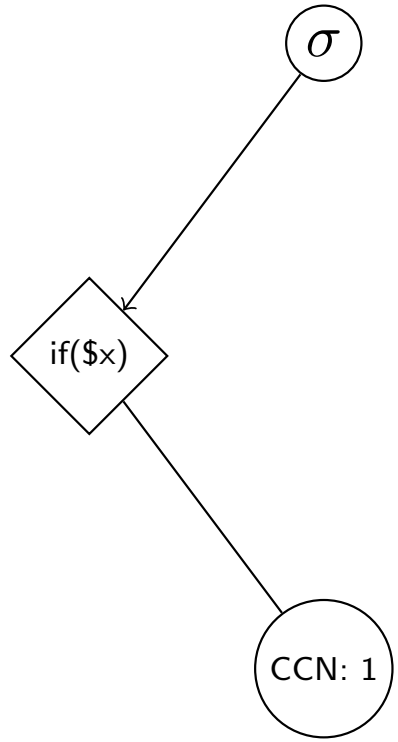
Cyclomatic Complexity


```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```



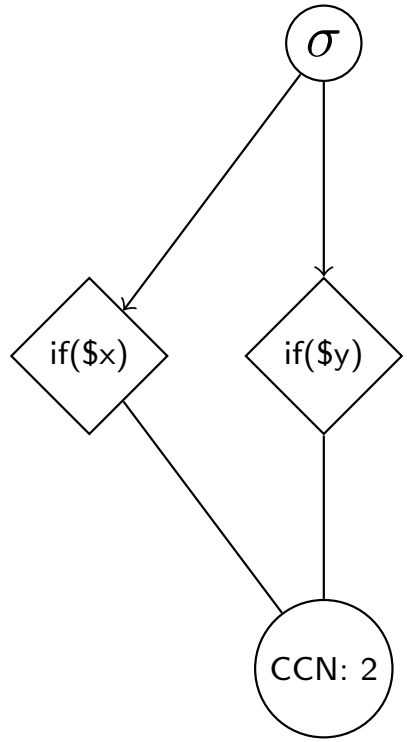
Cyclomatic Complexity

```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```



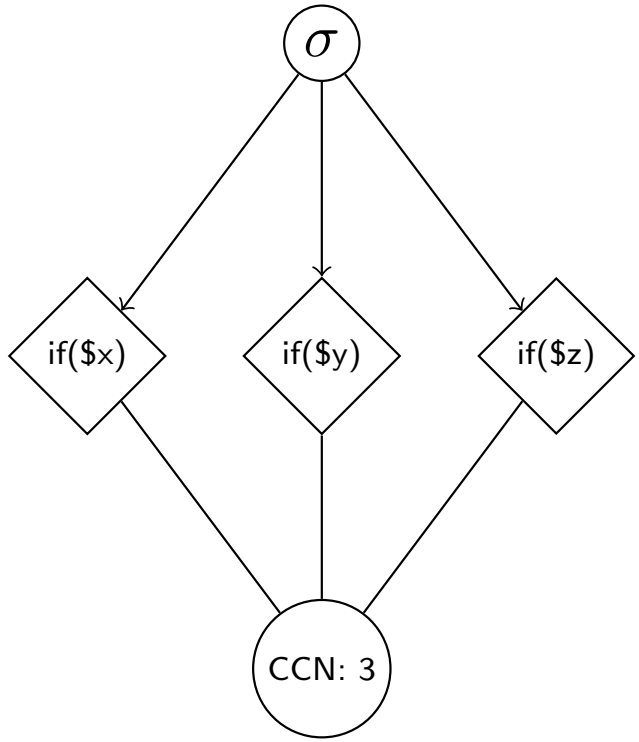
Cyclomatic Complexity

```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```



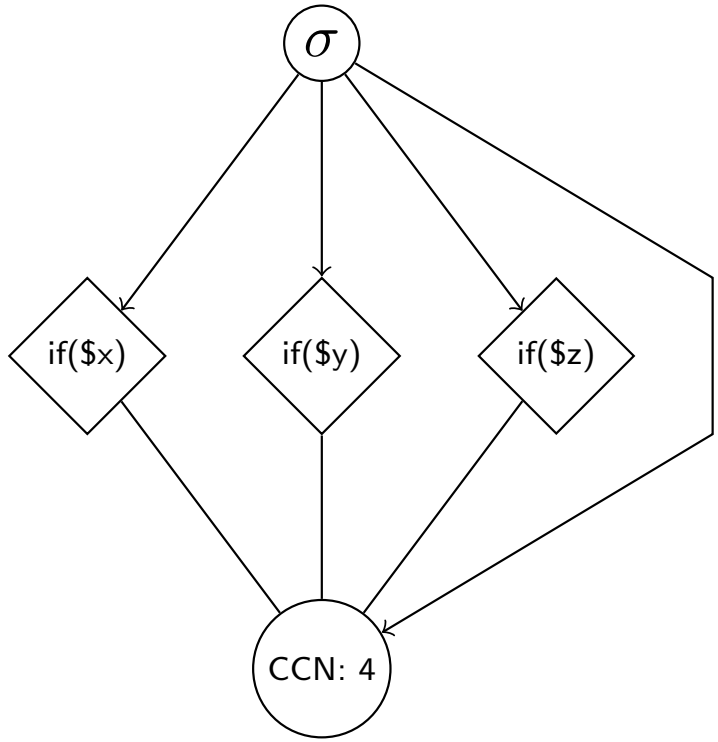
Cyclomatic Complexity

```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```



Cyclomatic Complexity

```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```



NPath Complexity

Wichtig

CCN == Anzahl der Verzweigungen

NPath Complexity

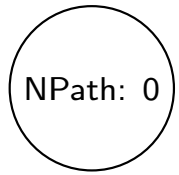
Wichtig

~~CCN == Anzahl der Verzweigungen~~

NPath == Anzahl der Ausführungspfade

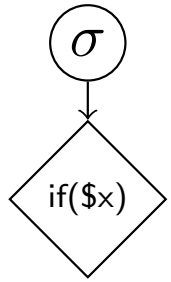
NPath Complexity

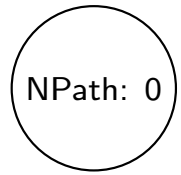

```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```



NPath Complexity

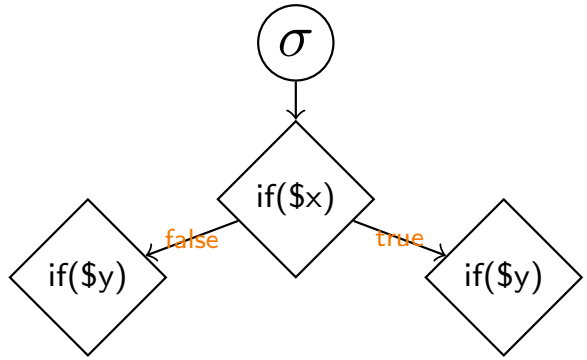
```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```

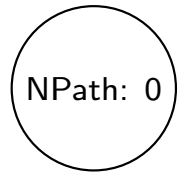




NPath Complexity

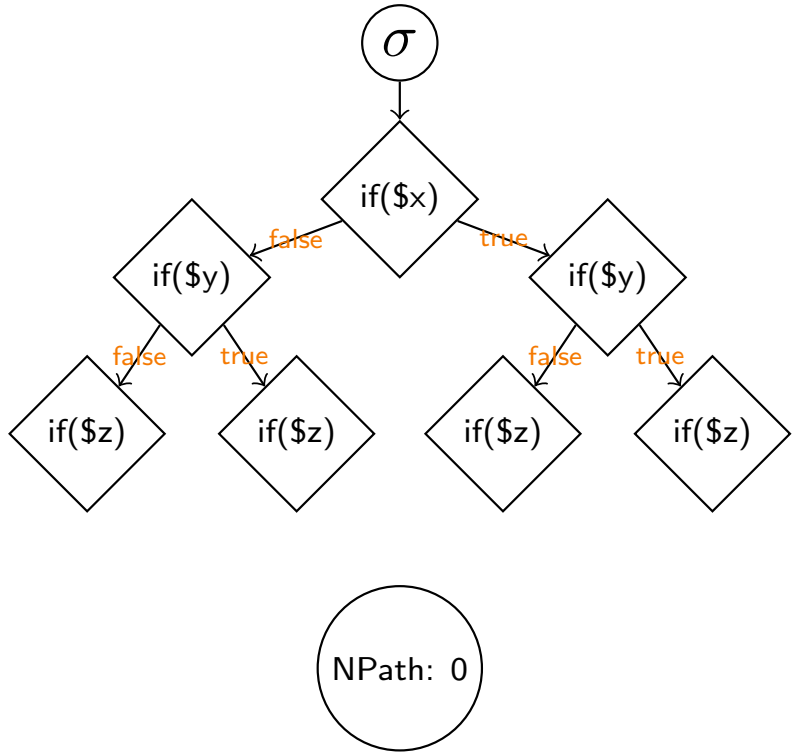
```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```





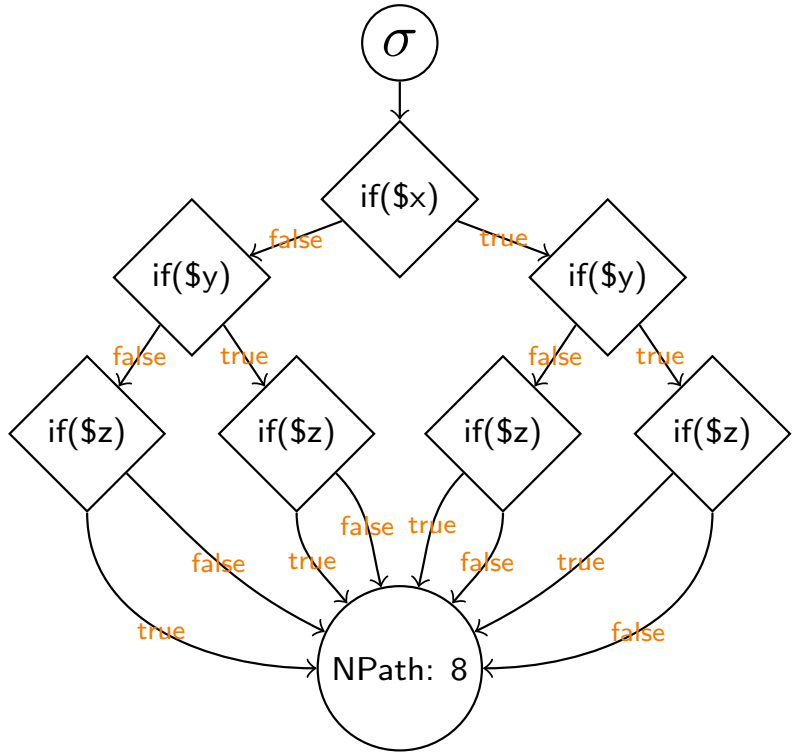
NPath Complexity

```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```



NPath Complexity

```
1
2   class Foo {
3       public function foo($x,$y,$z) {
4           if ($x) { $y -= $x; }
5           if ($y) { $z += $y * $x; }
6           if ($z) { echo $z; }
7       }
8   }
```



Grenzwerte in PHPMD

- Cyclomatic Complexity
  - 1-4: low, 5-7: medium, 8-10: high, 11+: hell
- NPath Complexity
  - 200: Ab hier wird es richtig schwer

Outline

Chidamber & Kemerer

- Weighted Methods per Class (WMC)
  - Summe der Komplexität aller Methoden
  - Grenzwert 20 - 50

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

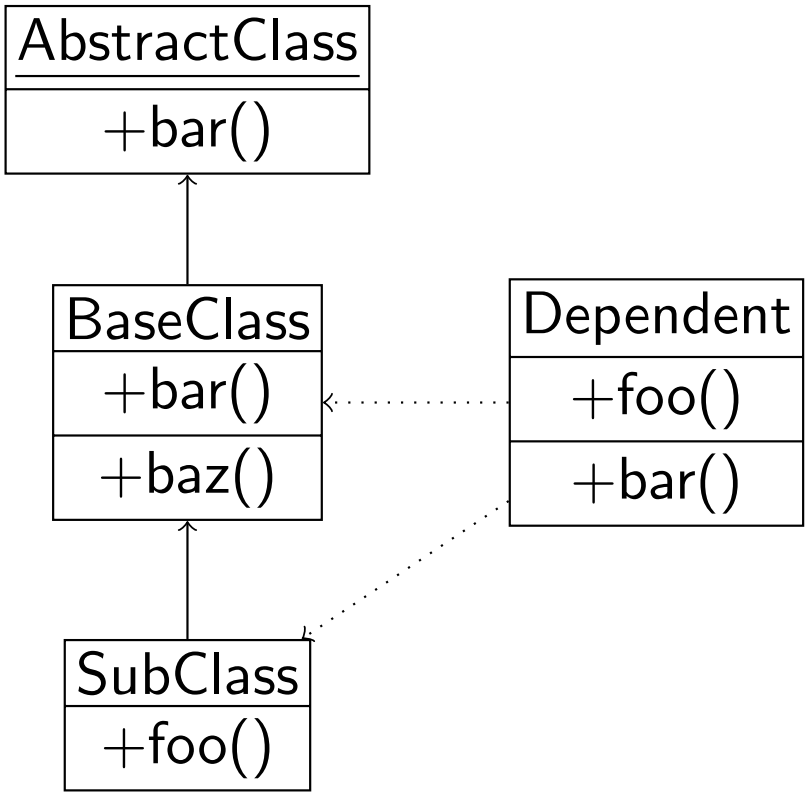
Chidamber & Kemerer

- 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

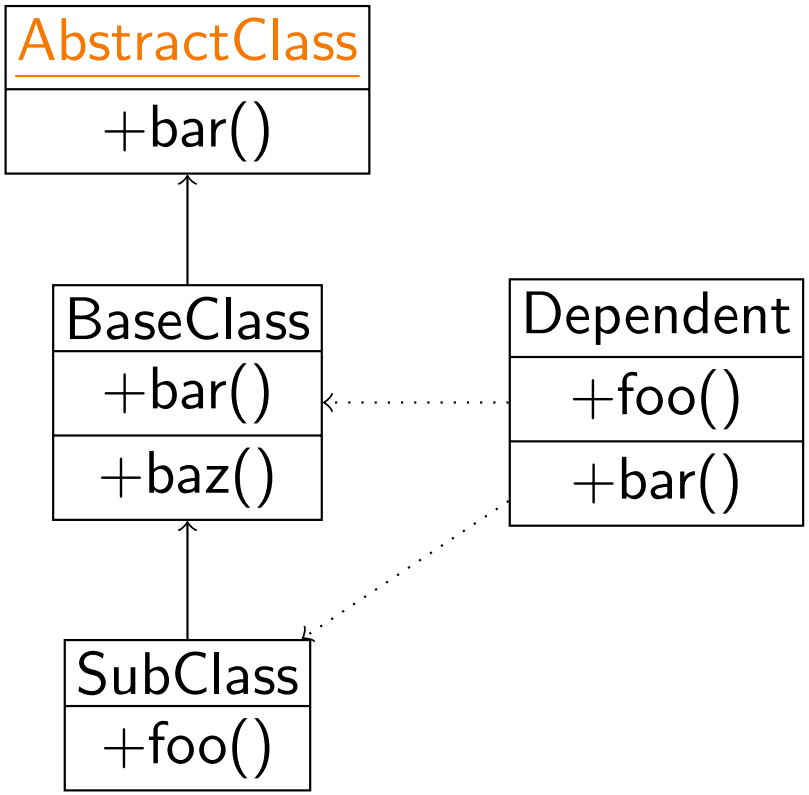
Chidamber & Kemerer

- 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
- Lack of Cohesion of Methods (LCOM)
  - Deplazierte Attribute und Methoden
  - Werte $> 1 ==$ Klasse besteht aus $N Komponenten

Chidamber & Kemerer Beispiel



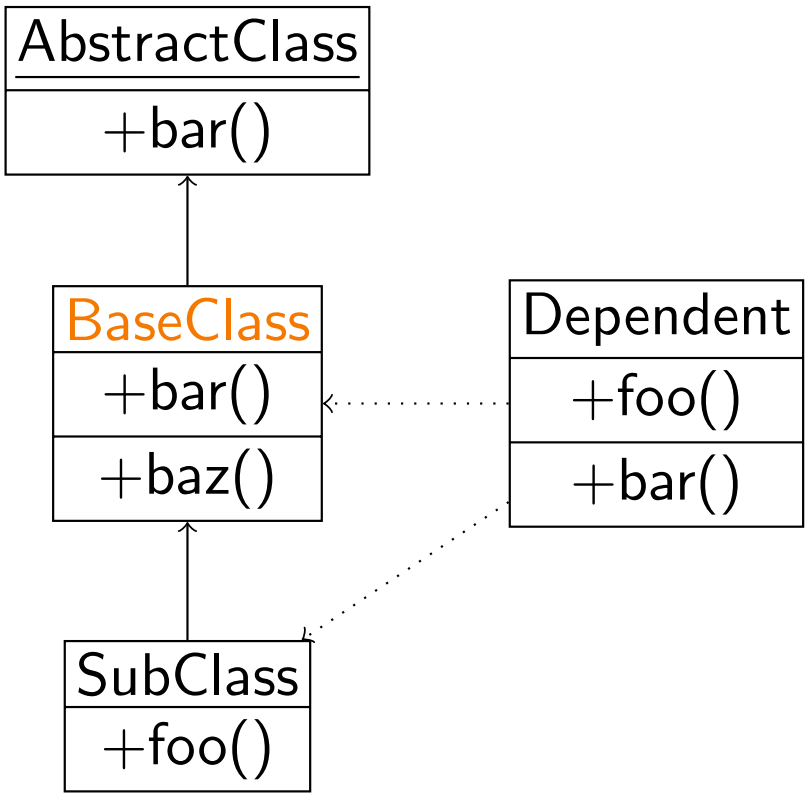
Chidamber & Kemerer Beispiel



- AbstractClass

WMC 0
DIT 0
NOC 1

Chidamber & Kemerer Beispiel



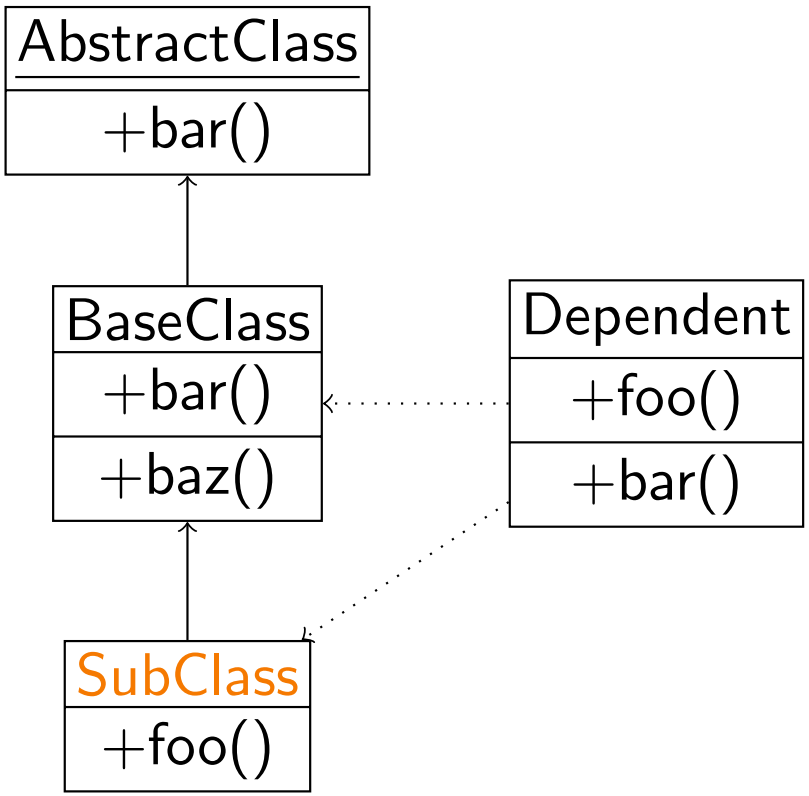
- AbstractClass

  WMC 0
  DIT 0
  NOC 1

- BaseClass

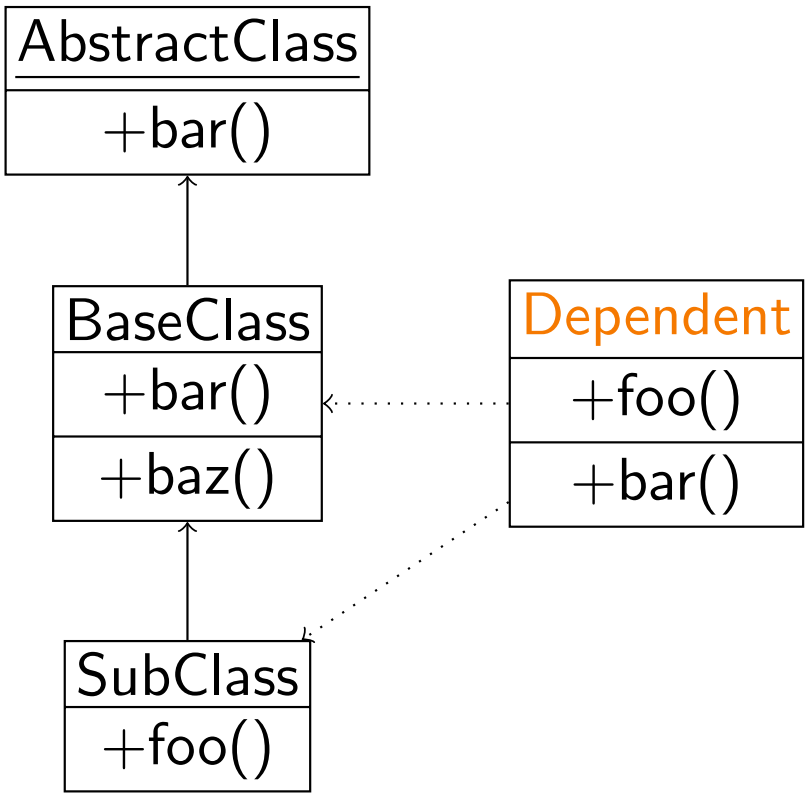
  WMC 2
  DIT 1
  NOC 1

Chidamber & Kemerer Beispiel



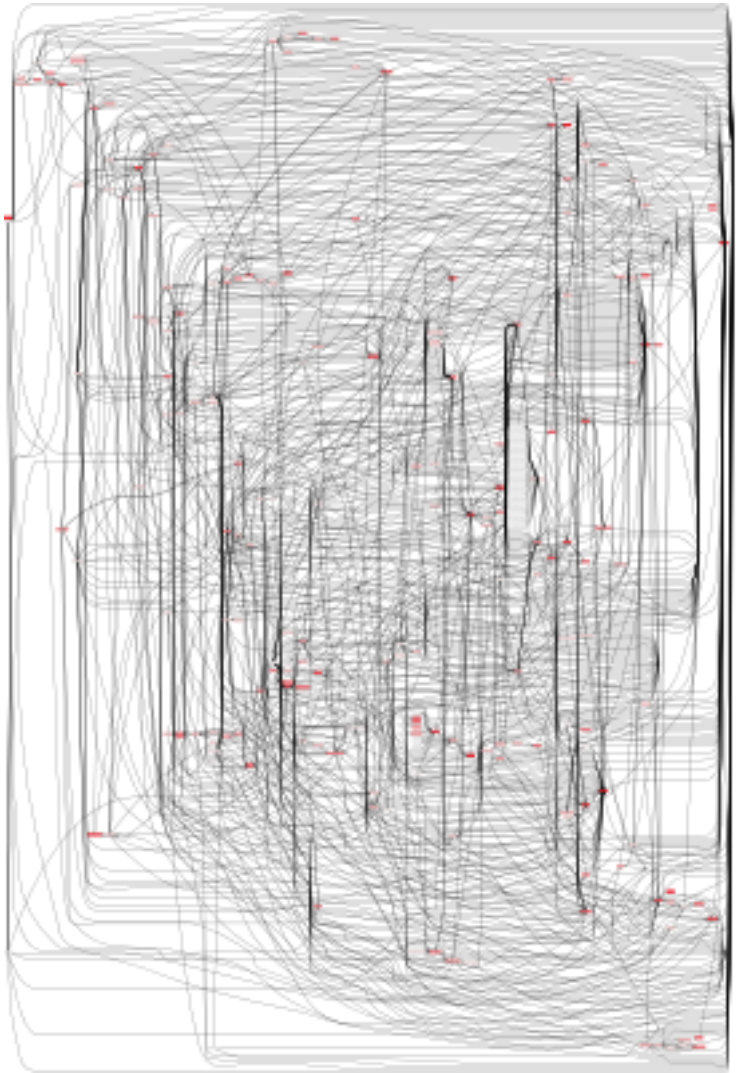
- AbstractClass
  - WMC 0
  - DIT 0
  - NOC 1
- BaseClass
  - WMC 2
  - DIT 1
  - NOC 1
- SubClass
  - WMC 1
  - DIT 2
  - NOC 0

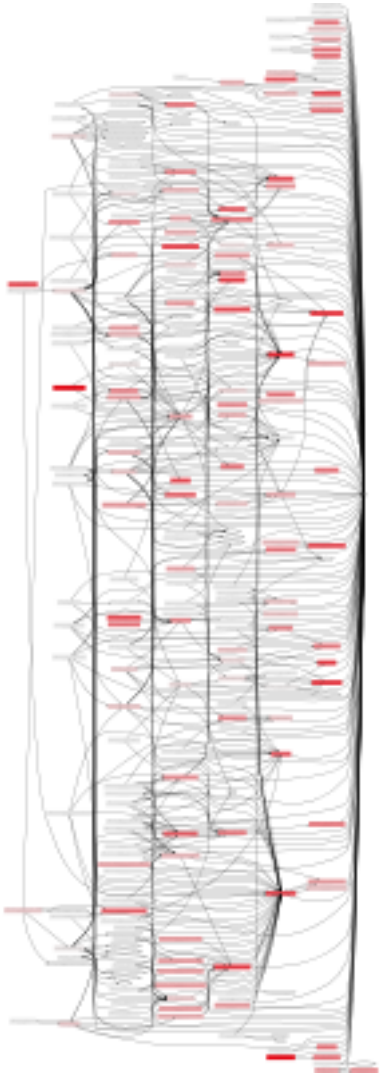
Chidamber & Kemerer Beispiel

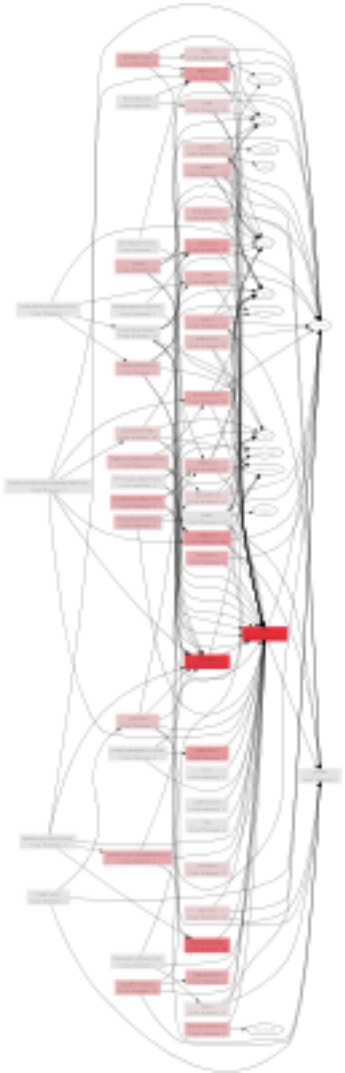


- AbstractClass
  - WMC 0
  - DIT 0
  - NOC 1
- BaseClass
  - WMC 2
  - DIT 1
  - NOC 1
- SubClass
  - WMC 1
  - DIT 2
  - NOC 0
- Dependent
  - WMC 2
  - DIT 0
  - NOC 0

Abhängigkeiten



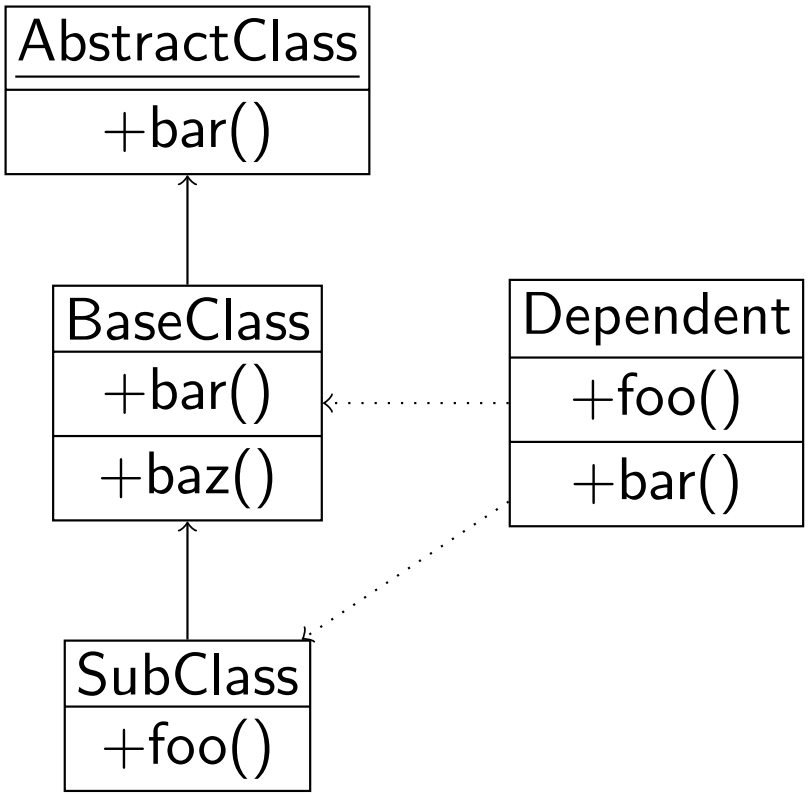




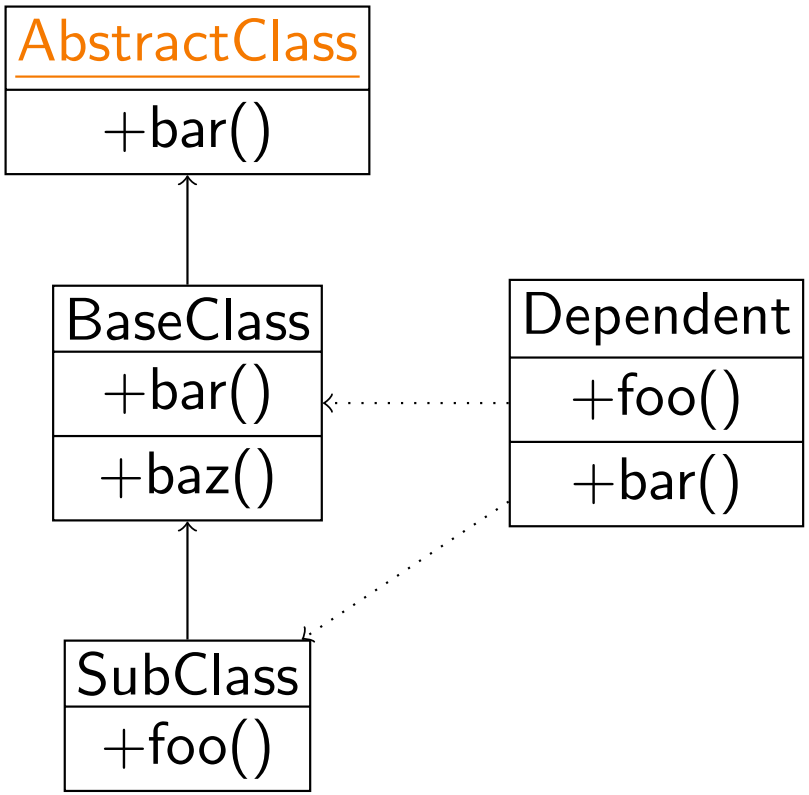
Afferent- & Efferent-Coupling

- Afferent Coupling (Ca)
  - Eingehende Abhängigkeiten anderer Komponenten
  - Großer Einfluss aus die Stabilität des Systems
- Efferent Coupling (Ce)
  - Ausgehende Abhängigkeiten:
    - Vererbung, Parameter, Exceptions, Allokationen
  - Abhängig von Stabilität anderer Komponenten

CE & CA Beispiel



CE & CA Beispiel

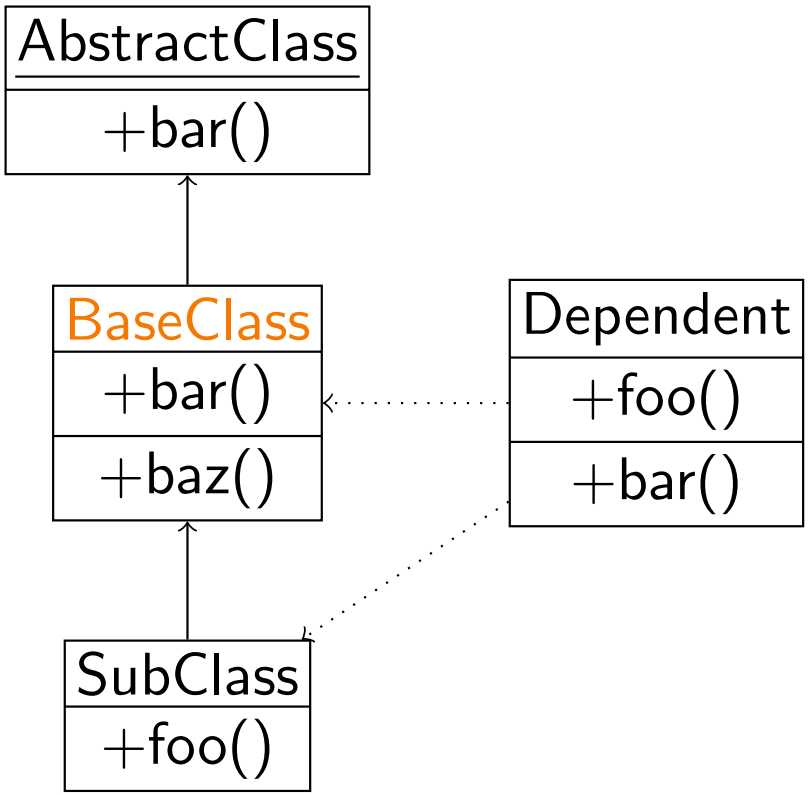


- AbstractClass

  Ce 0

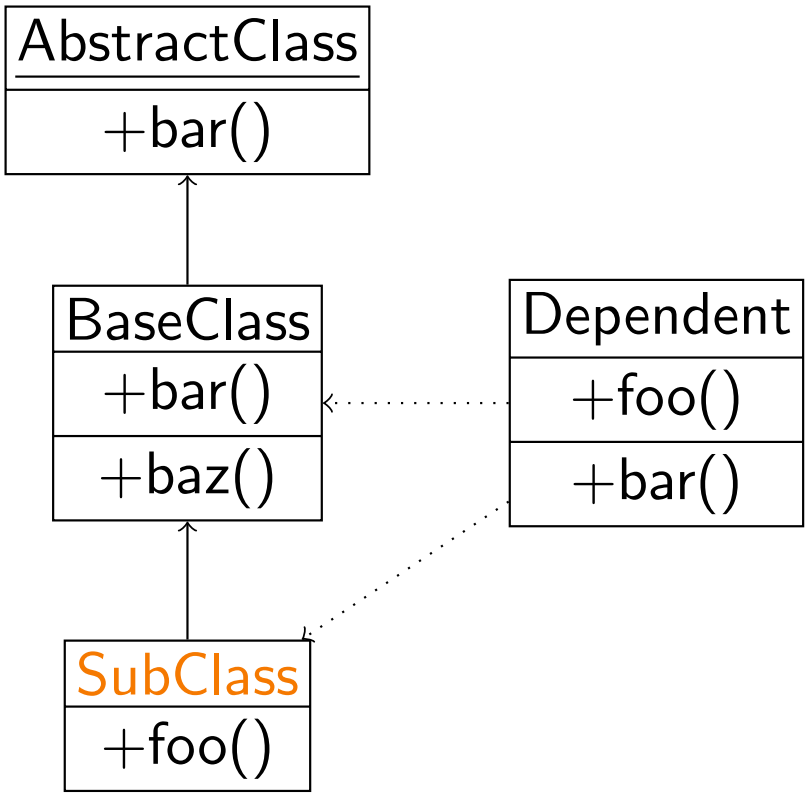
  Ca 1

CE & CA Beispiel



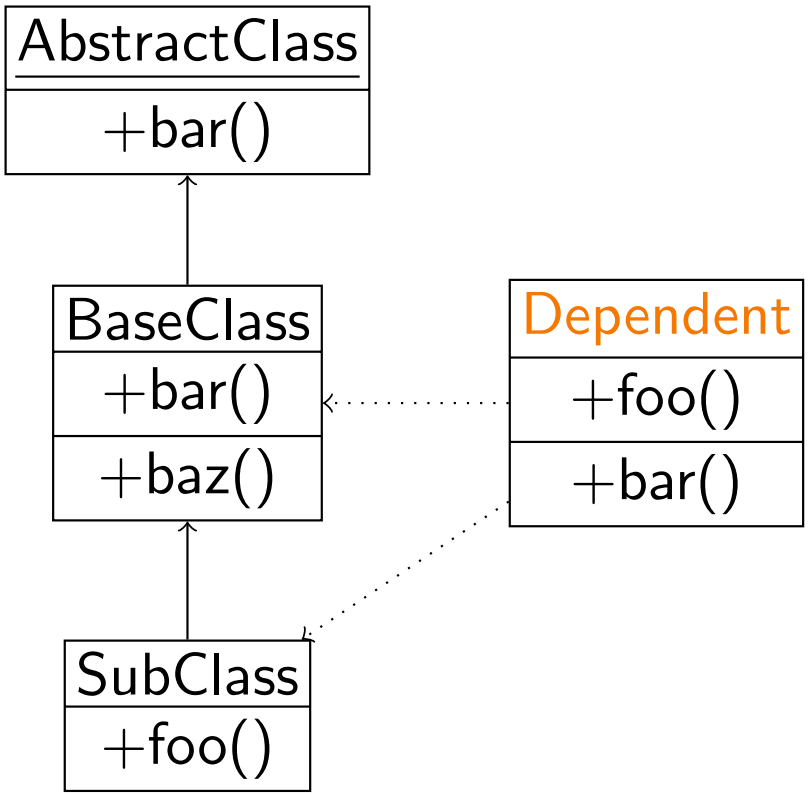
- AbstractClass
  - Ce 0
  - Ca 1
- BaseClass
  - Ce 1
  - Ca 2

CE & CA Beispiel



- AbstractClass
  - Ce 0
  - Ca 1
- BaseClass
  - Ce 1
  - Ca 2
- SubClass
  - Ce 1
  - Ca 1

CE & CA Beispiel



- AbstractClass
  - Ce 0
  - Ca 1
- BaseClass
  - Ce 1
  - Ca 2
- SubClass
  - Ce 1
  - Ca 1
- Dependent
  - Ce 2
  - Ca 0

Frei nach: Viele Wege führen nach Rom

- Häufig synonym verwendete Bezeichnungen für Ce & Ca

Frei nach: Viele Wege führen nach Rom

- Häufig synonym verwendete Bezeichnungen für Ce & Ca
  - FANIN := Ca

Frei nach: Viele Wege führen nach Rom

- Häufig synonym verwendete Bezeichnungen für Ce & Ca
  - FANIN := Ca
  - FANOUT := Ce

Frei nach: Viele Wege führen nach Rom

- Häufig synonym verwendete Bezeichnungen für Ce & Ca
  - FANIN := Ca
  - FANOUT := Ce
  - CBO := Ce

Frei nach: Viele Wege führen nach Rom

- Häufig synonym verwendete Bezeichnungen für Ce & Ca
  - FANIN := Ca
  - FANOUT := Ce
  - CBO := Ce
- Geänderte Semantik im laufe der Zeit

Frei nach: Viele Wege führen nach Rom

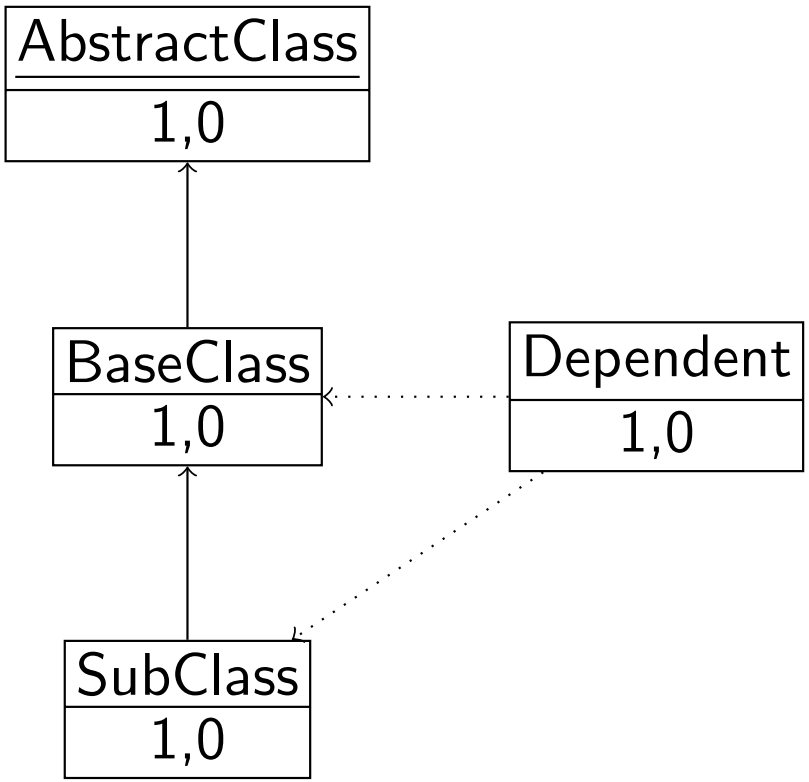
- Häufig synonym verwendete Bezeichnungen für Ce & Ca
  - FANIN := Ca
  - FANOUT := Ce
  - CBO := Ce
- Geänderte Semantik im laufe der Zeit
  - Ursprünglich: CBO = (Ce + Ca)

CodeRank

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

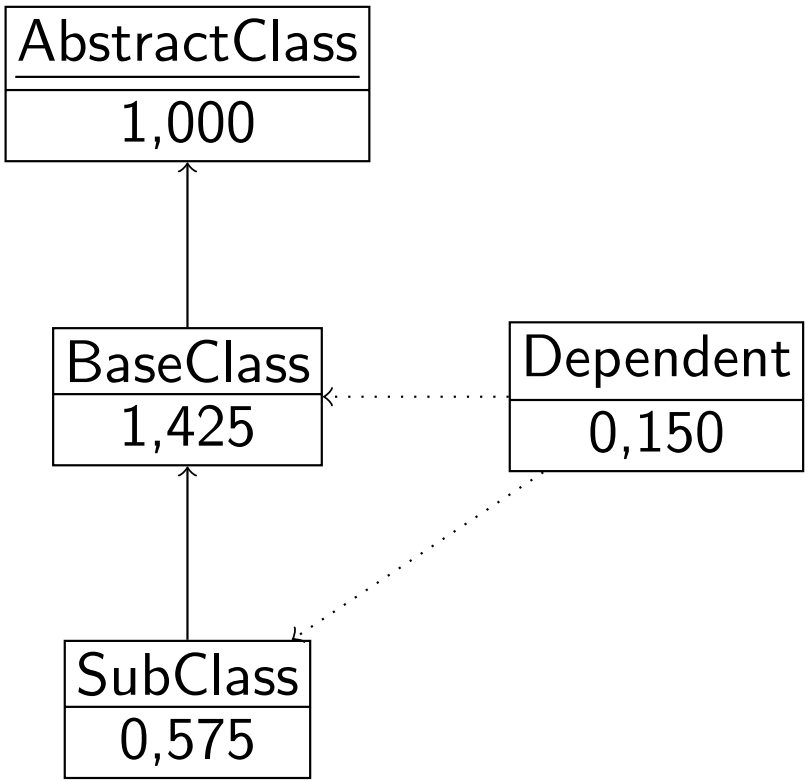
$$CR(\pi_i) = \sum_r r((1-d) + d \sum_r r(CR(\pi_r)/\rho_r))$$

CodeRank



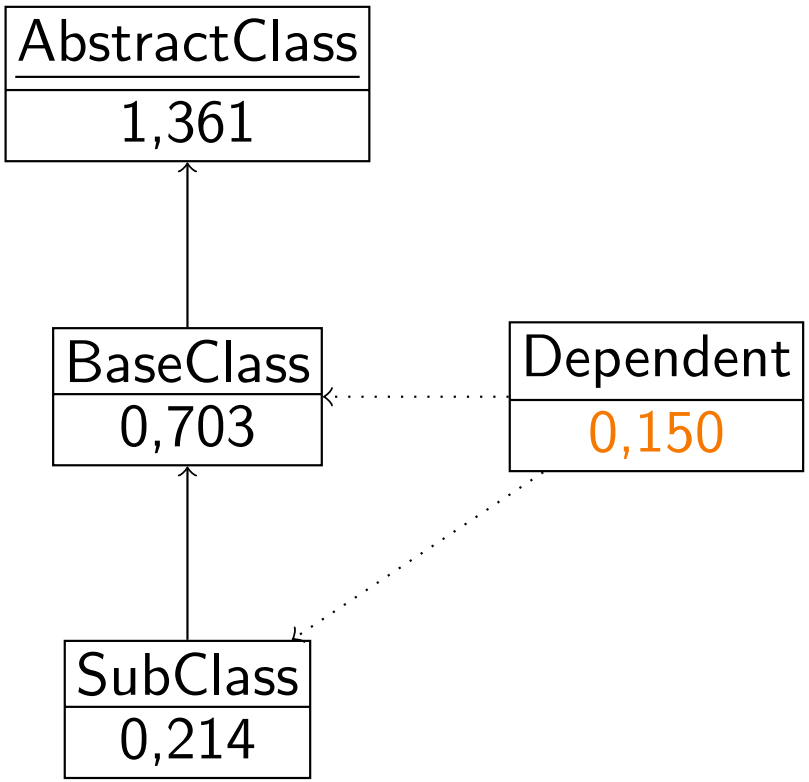
- Iteration: 0

CodeRank



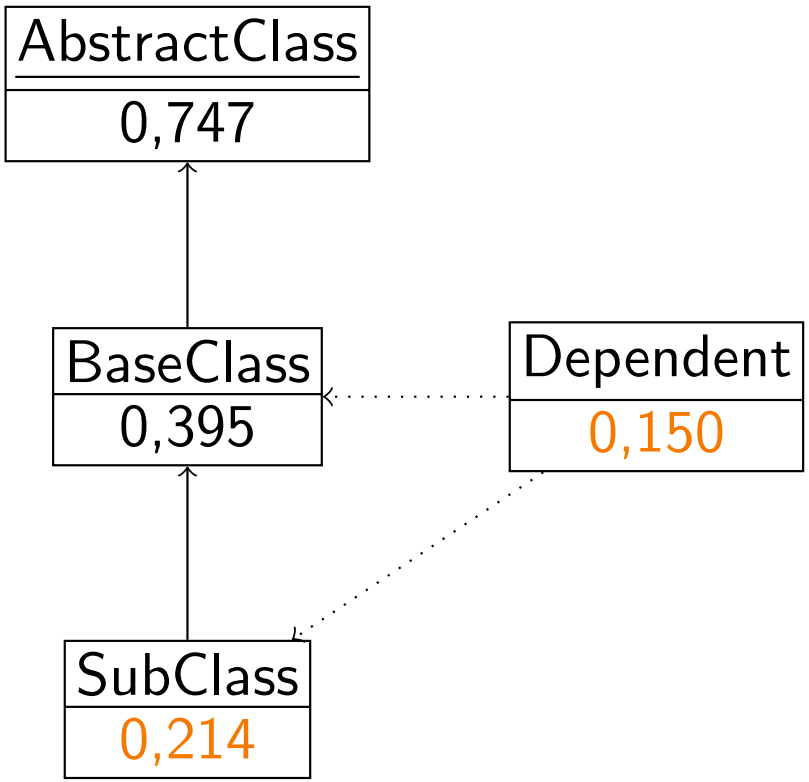
- Iteration: 1

CodeRank



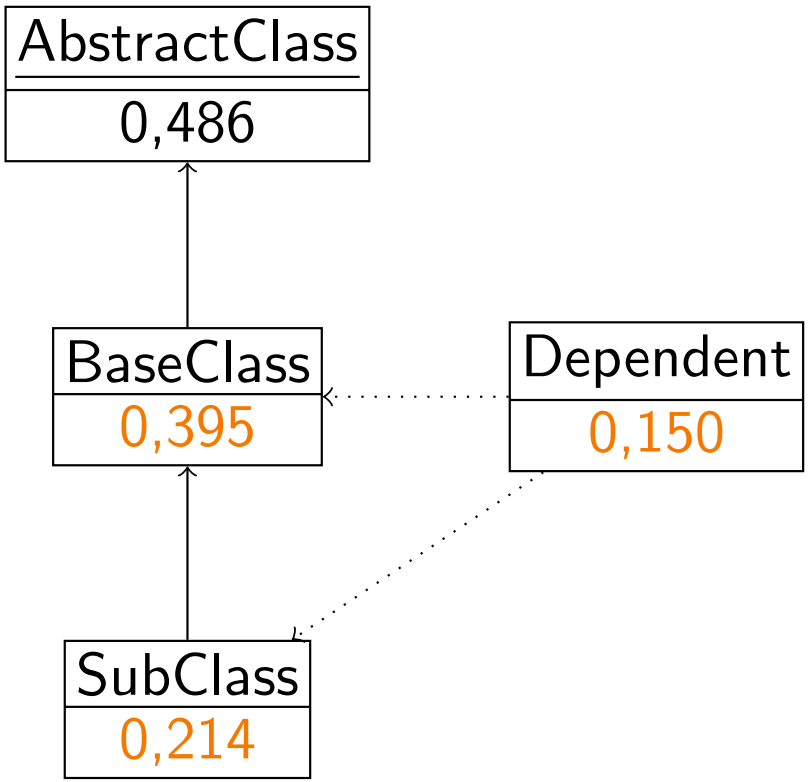
- Iteration: 2

CodeRank



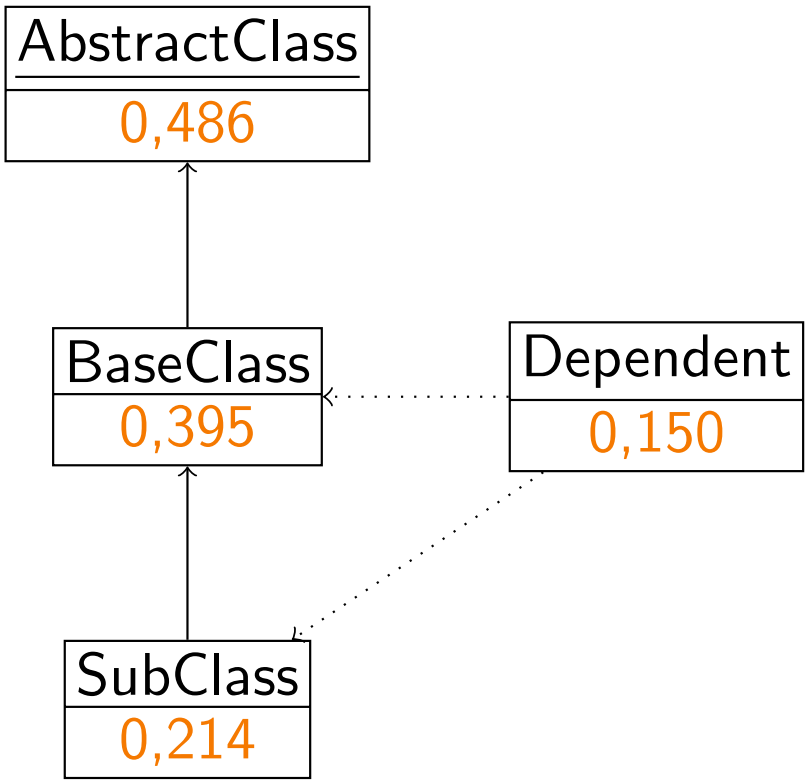
- Iteration: 3

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

Grenzwerte

Was bedeuten aber Werte wie 4, 8 oder 0,486 nun genau?

Grenzwerte

Was bedeuten aber Werte wie 4, 8 oder 0,486 nun genau?

- Die Zahlen allein sagen noch nichts aus

Grenzwerte

Was bedeuten aber Werte wie 4, 8 oder 0,486 nun genau?

- Die Zahlen allein sagen noch nichts aus
- Für jede Beurteilung benötigt man einen Kontext

Grenzwerte

Was bedeuten aber Werte wie 4, 8 oder 0,486 nun genau?

- Die Zahlen allein sagen noch nichts aus
- Für jede Beurteilung benötigt man einen Kontext
- Im Fall von Metriken sind das Grenzwerte

Grenzwerte

Was bedeuten aber Werte wie 4, 8 oder 0,486 nun genau?

- Die Zahlen allein sagen noch nichts aus
- Für jede Beurteilung benötigt man einen Kontext
- Im Fall von Metriken sind das Grenzwerte

Aber woher bekommt man sinnvolle Grenzwerte?

Statistische Auswertung I

- Analysiere alle existierenden Projekte. . .

Statistische Auswertung I

- Analysiere alle existierenden Projekte. . .
  - . . . um einen Überblick über Mittelwerte zu bekommen

Statistische Auswertung I

- Analysiere alle existierenden Projekte...
  - ...um einen Überblick über Mittelwerte zu bekommen
  - ...um potentielle Grenzwerte/Ausreißer zu identifizieren

Statistische Auswertung I

- Analysiere alle existierenden Projekte...
  - ...um einen Überblick über Mittelwerte zu bekommen
  - ...um potentielle Grenzwerte/Ausreißer zu identifizieren
- Analysiere zusätzlich verwendete Frameworks...

Statistische Auswertung I

- Analysiere alle existierenden Projekte...
  - ...um einen Überblick über Mittelwerte zu bekommen
  - ...um potentielle Grenzwerte/Ausreißer zu identifizieren
- Analysiere zusätzlich verwendete Frameworks...
  - ...hierdurch werden die ermittelten Werte heterogener

Statistische Auswertung II

- Einfache Berechnung

Statistische Auswertung II

- Einfache Berechnung
  - Mittelwert für einzelne Metriken (**AVG**)

Statistische Auswertung II

- Einfache Berechnung
  - Mittelwert für einzelne Metriken (**AVG**)
  - Standardabweichung vom Mittelwert (**STDEV**)

Statistische Auswertung II

- Einfache Berechnung
  - Mittelwert für einzelne Metriken (**AVG**)
  - Standardabweichung vom Mittelwert (**STDEV**)
    - So Statistikgedöns

Statistische Auswertung II

- Einfache Berechnung
  - Mittelwert für einzelne Metriken (**AVG**)
  - Standardabweichung vom Mittelwert (**STDEV**)
    - So Statistikgedöns
    - 3. Semester Informatik Studium ;-]

Statistische Auswertung II

- Einfache Berechnung
  - Mittelwert für einzelne Metriken (**AVG**)
  - Standardabweichung vom Mittelwert (**STDEV**)
    - So Statistikgedöns
    - 3. Semester Informatik Studium ;-]
- Grenzwert-Berechnung

Statistische Auswertung II

- Einfache Berechnung
  - Mittelwert für einzelne Metriken (**AVG**)
  - Standardabweichung vom Mittelwert (**STDEV**)
    - So Statistikgedöns
    - 3. Semester Informatik Studium ;-]
- Grenzwert-Berechnung
  - Untere Grenze: `AVG - STDEV`

Statistische Auswertung II

- Einfache Berechnung
  - Mittelwert für einzelne Metriken (**AVG**)
  - Standardabweichung vom Mittelwert (**STDEV**)
    - So Statistikgedöns
    - 3. Semester Informatik Studium ;-]
- Grenzwert-Berechnung
  - Untere Grenze: `AVG - STDEV`
  - Obere Grenze: `AVG + STDEV`

Statistische Auswertung II

- Einfache Berechnung
  - Mittelwert für einzelne Metriken (**AVG**)
  - Standardabweichung vom Mittelwert (**STDEV**)
    - So Statistikgedöns
    - 3. Semester Informatik Studium ;-]
- Grenzwert-Berechnung
  - Untere Grenze: `AVG - STDEV`
  - Obere Grenze: `AVG + STDEV`
  - Grenzwertig: `1,5 * (AVG + STDEV)`

Grenzwerte

- First Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.19	0.21	0.22	0.33
LOC/NOM	10.89	14.55	18.20	27.30
NOM/NOC	6.91	7.51	8.11	12.17
NOC/NOP	6.27	8.10	9.93	14.89

Grenzwerte

- First Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.19	0.21	0.22	0.33
LOC/NOM	10.89	14.55	18.20	27.30
NOM/NOC	6.91	7.51	8.11	12.17
NOC/NOP	6.27	8.10	9.93	14.89

- Second Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.18	0.20	0.22	0.32
LOC/NOM	10.55	11.98	13.41	20.11
NOM/NOC	5.07	5.96	6.85	10.28
NOC/NOP	3.69	4.23	4.78	7.17

Grenzwerte

- First Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.19	0.21	0.22	0.33
LOC/NOM	10.89	14.55	18.20	27.30
NOM/NOC	6.91	7.51	8.11	12.17
NOC/NOP	6.27	8.10	9.93	14.89

- Second Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.18	0.20	0.22	0.32
LOC/NOM	10.55	11.98	13.41	20.11
NOM/NOC	5.07	5.96	6.85	10.28
NOC/NOP	3.69	4.23	4.78	7.17

Grenzwerte

- First Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.19	0.21	0.22	0.33
LOC/NOM	10.89	14.55	18.20	27.30
NOM/NOC	6.91	7.51	8.11	12.17
NOC/NOP	6.27	8.10	9.93	14.89

- Second Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.18	0.20	0.22	0.32
LOC/NOM	10.55	11.98	13.41	20.11
NOM/NOC	5.07	5.96	6.85	10.28
NOC/NOP	3.69	4.23	4.78	7.17

Grenzwerte

- First Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.19	0.21	0.22	0.33
LOC/NOM	10.89	14.55	18.20	27.30
NOM/NOC	6.91	7.51	8.11	12.17
NOC/NOP	6.27	8.10	9.93	14.89

- Second Generation PHP-Frameworks

Metrik	Low	Avg	High	Outliner
CCN/LOC	0.18	0.20	0.22	0.32
LOC/NOM	10.55	11.98	13.41	20.11
NOM/NOC	5.07	5.96	6.85	10.28
NOC/NOP	3.69	4.23	4.78	7.17

Der gesunde Menschenverstand

- Manche Grenzwerte erschließen sich direkt

Der gesunde Menschenverstand

- Manche Grenzwerte erschließen sich direkt
  - Beispiel: Tiefe von Kontrollstrukturen

Der gesunde Menschenverstand

- Manche Grenzwerte erschließen sich direkt
  - Beispiel: Tiefe von Kontrollstrukturen

```
if ( $x ) {




}
```

Der gesunde Menschenverstand

- Manche Grenzwerte erschließen sich direkt
  - Beispiel: Tiefe von Kontrollstrukturen

```
if ( $x ) {
    if ( $y == 23 ) {









    }
}
```

Der gesunde Menschenverstand

- Manche Grenzwerte erschließen sich direkt
  - Beispiel: Tiefe von Kontrollstrukturen

```
if ( $x ) {
    if ( $y == 23 ) {
        foreach ( $z as $a ) {









        }
    }
}
```

Der gesunde Menschenverstand

- Manche Grenzwerte erschließen sich direkt
  - Beispiel: Tiefe von Kontrollstrukturen

```
if ( $x ) {
    if ( $y == 23 ) {
        foreach ( $z as $a ) {
            if ( $a != 42 ) {
                $foo = ( $bar ?  $bar :  23 );
            } else {
                $bar = ( $foo ?  $foo :  42 );
            }
        }
    }
}
```

Der gesunde Menschenverstand

- Manche Grenzwerte erschließen sich direkt
  - Beispiel: Tiefe von Kontrollstrukturen

WTF?

Ganz wichtig

Grenzwerte sind immer Ermessenssache

Ganz wichtig

Grenzwerte sind immer Ermessenssache

Es gibt einfach kein Schwarz & Weiß

Outline

Einen Überblick verschaffen

- Wir haben jetzt einen schier unendlichen Fundus an Metriken

Einen Überblick verschaffen

- Wir haben jetzt einen schier unendlichen Fundus an Metriken
  - **Aber...**

Einen Überblick verschaffen

- Wir haben jetzt einen schier unendlichen Fundus an Metriken
  - **Aber...**
    - ... was mache ich jetzt damit?

Einen Überblick verschaffen

- Wir haben jetzt einen schier unendlichen Fundus an Metriken
  - **Aber...**
    - ... was mache ich jetzt damit?
    - ... welche Metriken helfen mir weiter?

Einen Überblick verschaffen

- Wir haben jetzt einen schier unendlichen Fundus an Metriken
  - **Aber...**
    - ... was mache ich jetzt damit?
    - ... welche Metriken helfen mir weiter?
    - ... wie fange ich an ein Projekt zu beurteilen?

Metriken kombinieren

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

Metriken kombinieren

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

- LOC: 300; CCN: 42; NOC: 5; NOM: 15

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

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

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

Metriken kombinieren

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

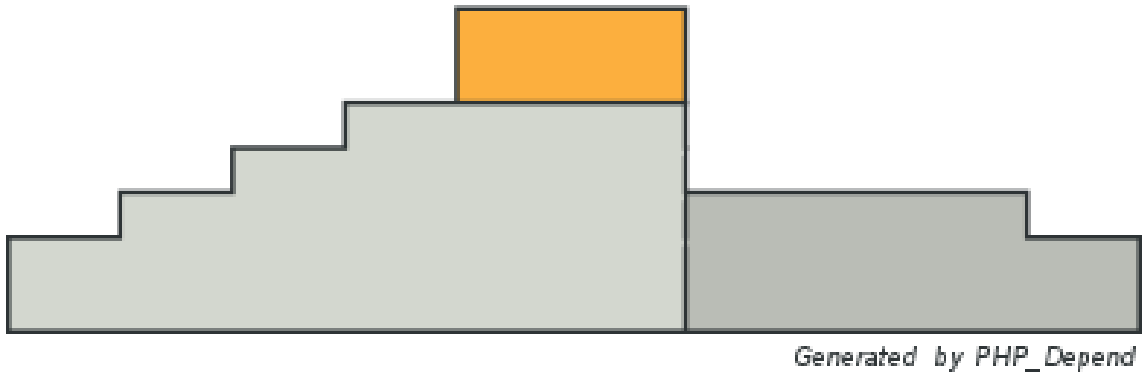
- LOC: 300; CCN: 42; NOC: 5; NOM: 15
  - CCN / LOC = 0,14
    - Jede sechste Zeile eine Kontrollstruktur
  - LOC / NOC = 60
    - Primär prozedural oder große Klassen
  - LOC / NOM = 20
    - Große Methoden / Funktionen
  - CCN / NOM = 2,8
    - Übermäßig komplexe Methoden / Funktionen

Auswertung

Visualisierung

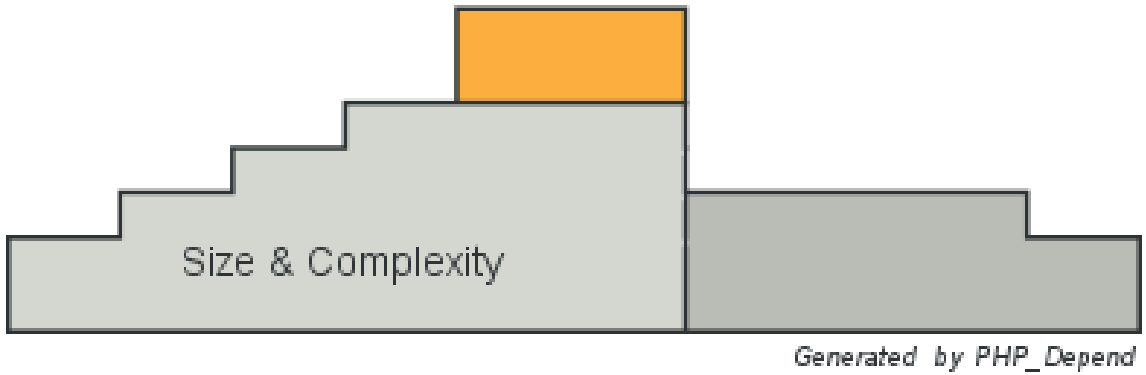
Overview Pyramid

- Wir brauchen eine übersichtliche Visualisierung



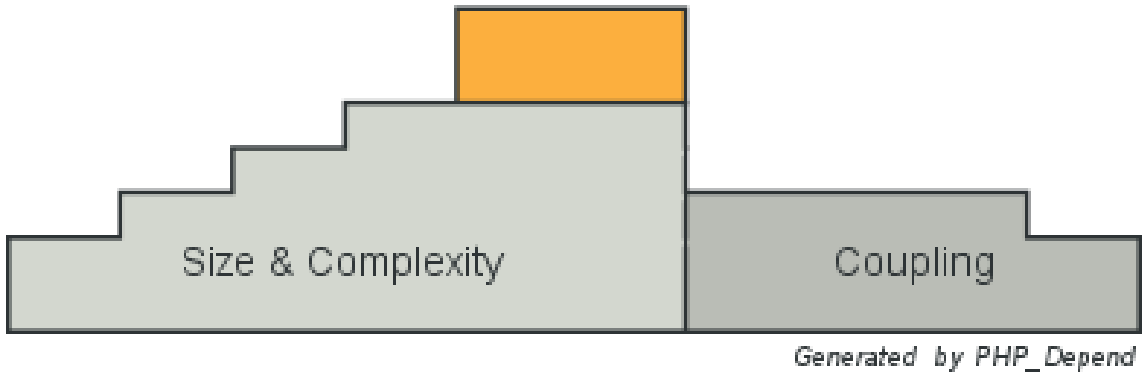
Overview Pyramid

- Wir brauchen eine übersichtliche Visualisierung
  - Bestehend aus drei unterschiedlichen Metrik-Typen



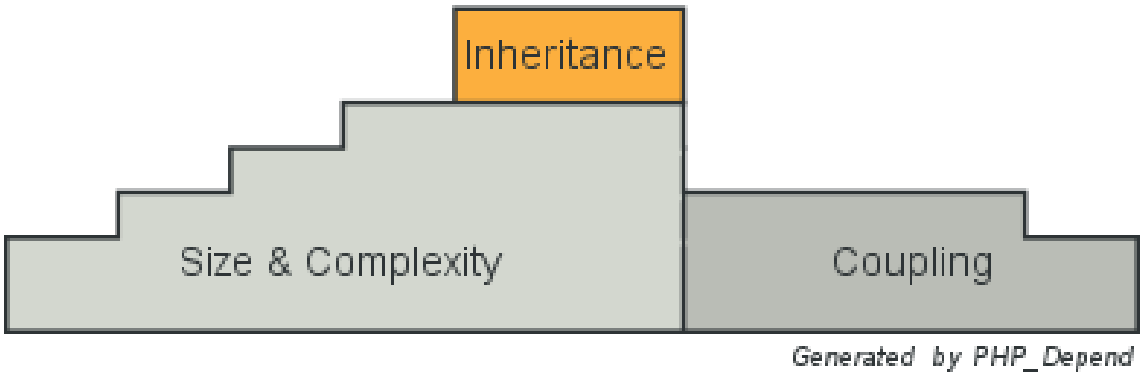
Overview Pyramid

- Wir brauchen eine übersichtliche Visualisierung
  - Bestehend aus drei unterschiedlichen Metrik-Typen



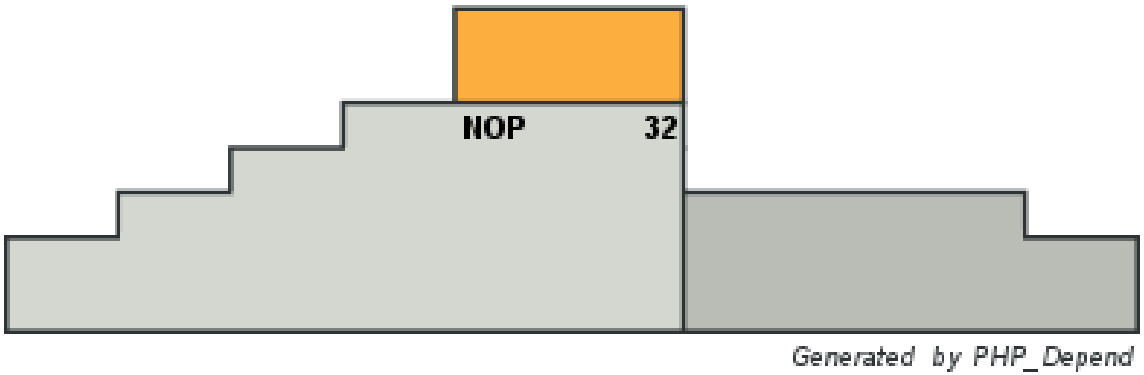
Overview Pyramid

- Wir brauchen eine übersichtliche Visualisierung
  - Bestehend aus drei unterschiedlichen Metrik-Typen



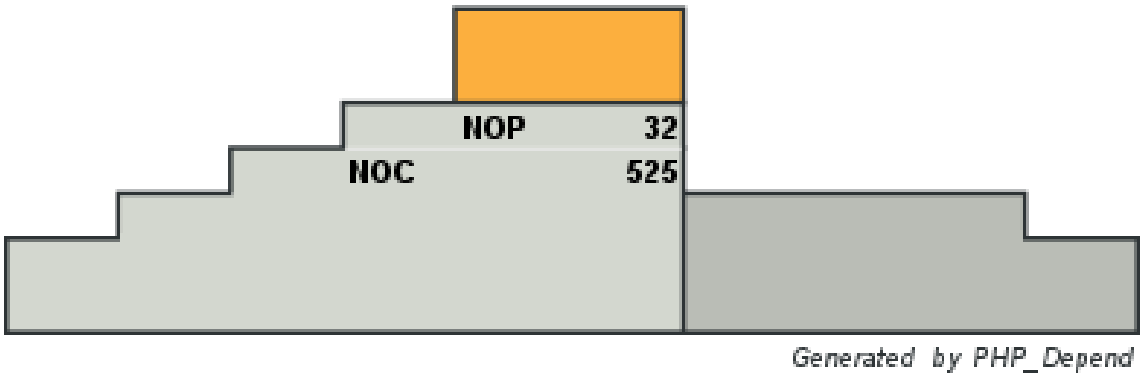
Overview Pyramid

- Umfang/Komplexität
  - **N**umber **O**f **P**ackages



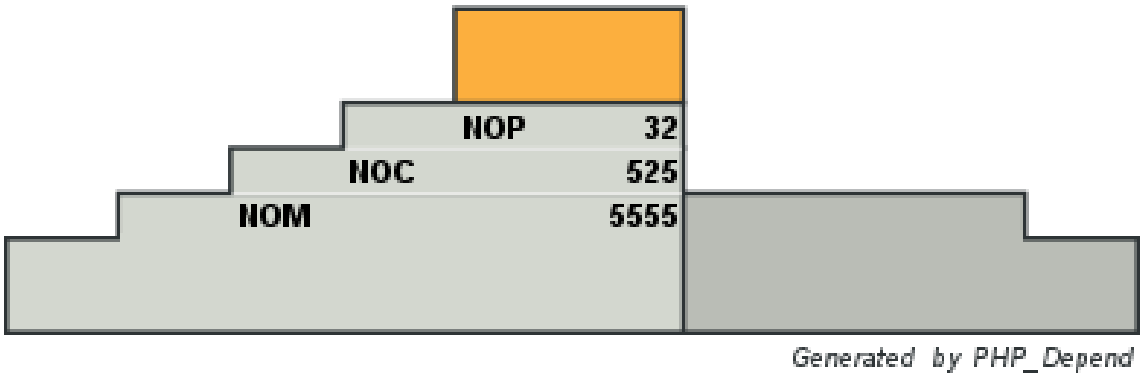
Overview Pyramid

- Umfang/Komplexität
  - **N**umber **O**f **C**lasses



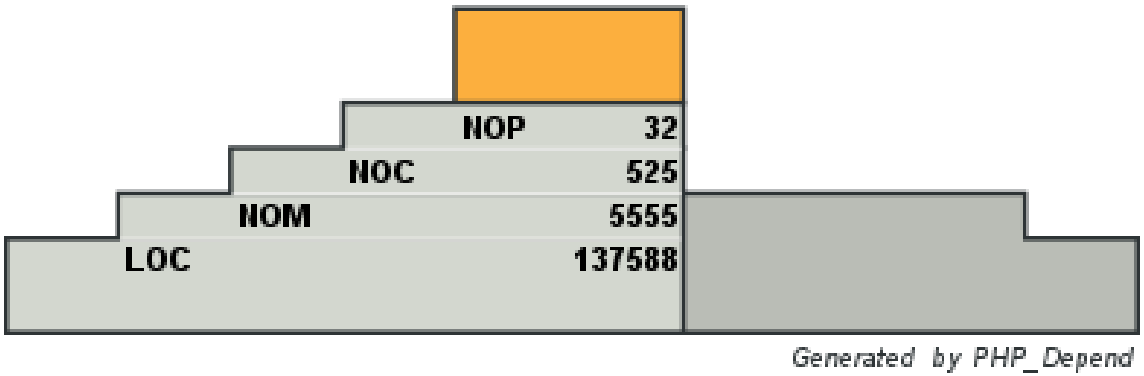
Overview Pyramid

- Umfang/Komplexität
  - **N**umber **O**f **M**ethods



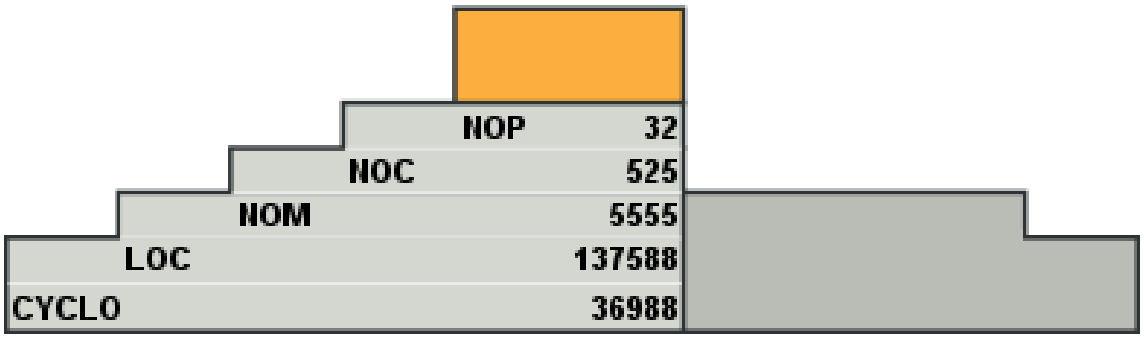
Overview Pyramid

- Umfang/Komplexität
  - **L**ines **O**f **C**ode



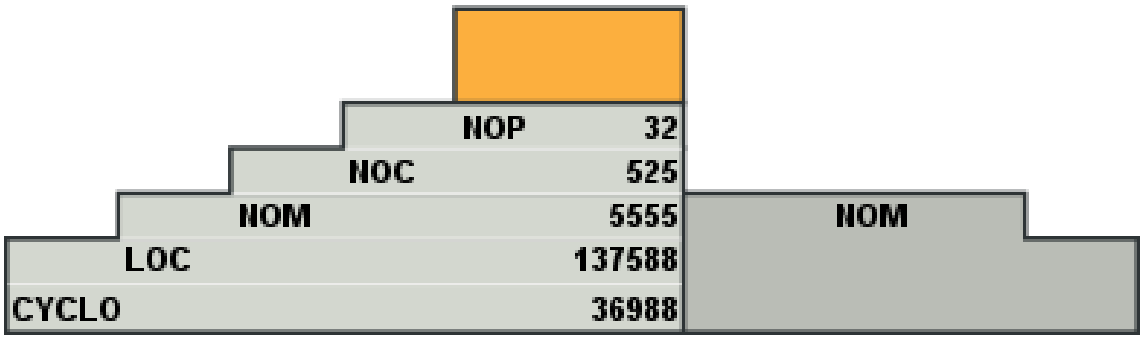
Overview Pyramid

- Umfang/Komplexität
  - **Cyclo**matic Complexity



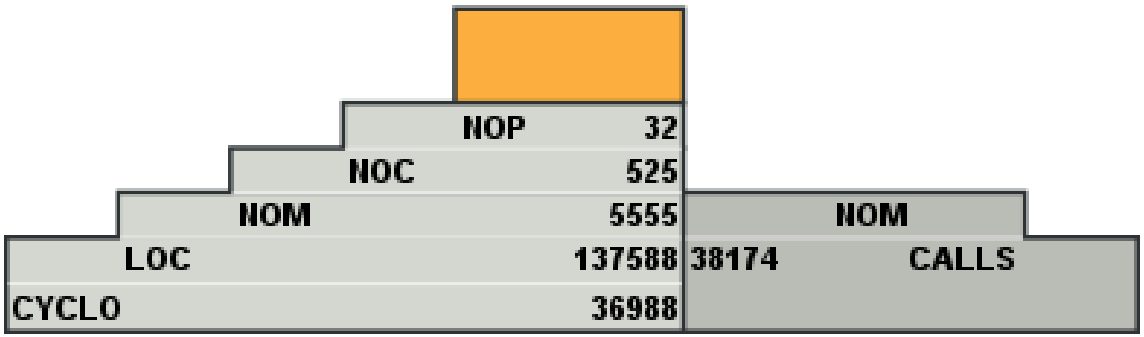
Overview Pyramid

- Koppelung
  - **N**umber **O**f **M**ethods



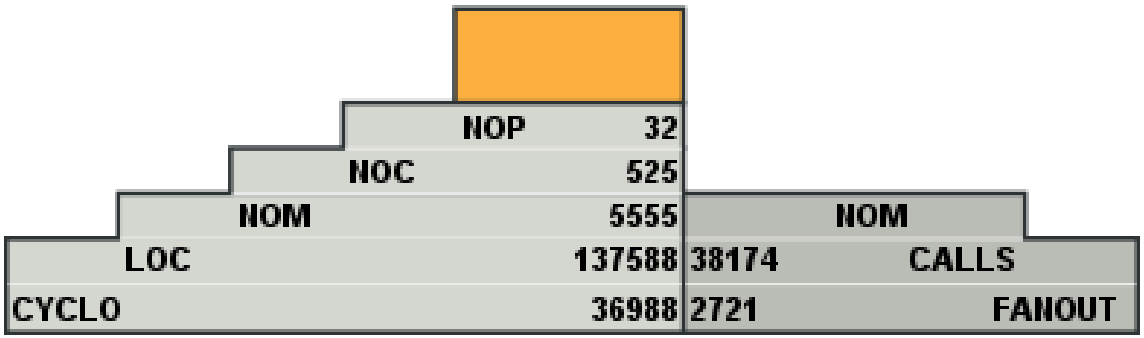
Overview Pyramid

- Koppelung
  - Number Of Operation **Calls**



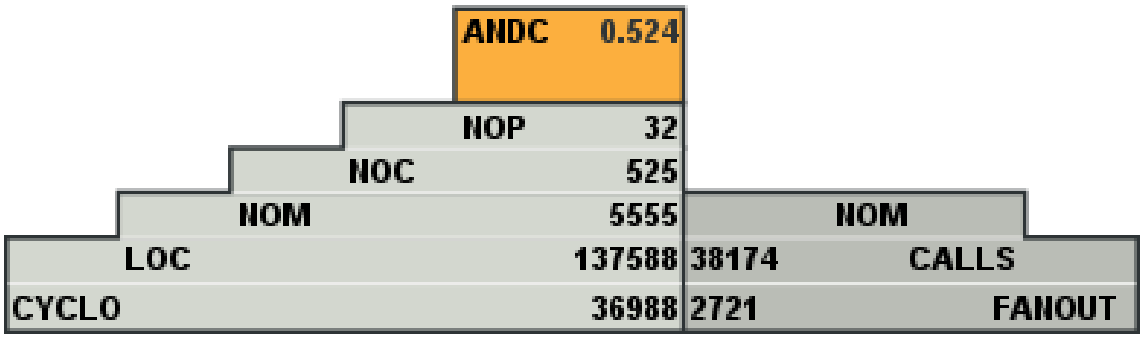
Overview Pyramid

- Koppelung
  - Number Of Called Classes



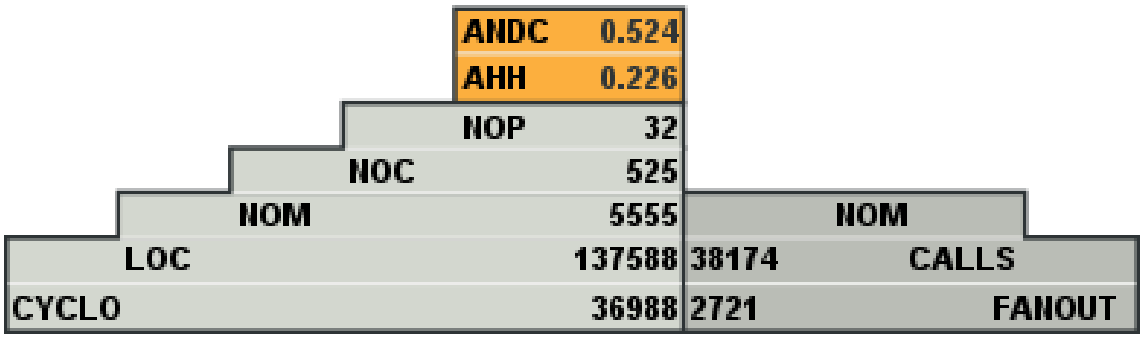
Overview Pyramid

- Vererbung
  - **A**verage **N**umber of **D**erived **C**lasses



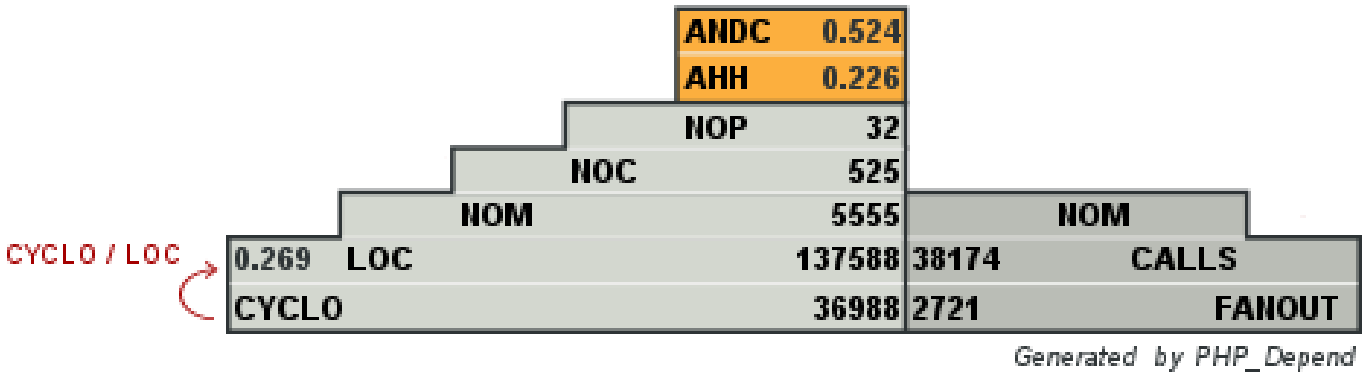
Overview Pyramid

- Vererbung
  - **A**verage **H**ierarchy **H**eight



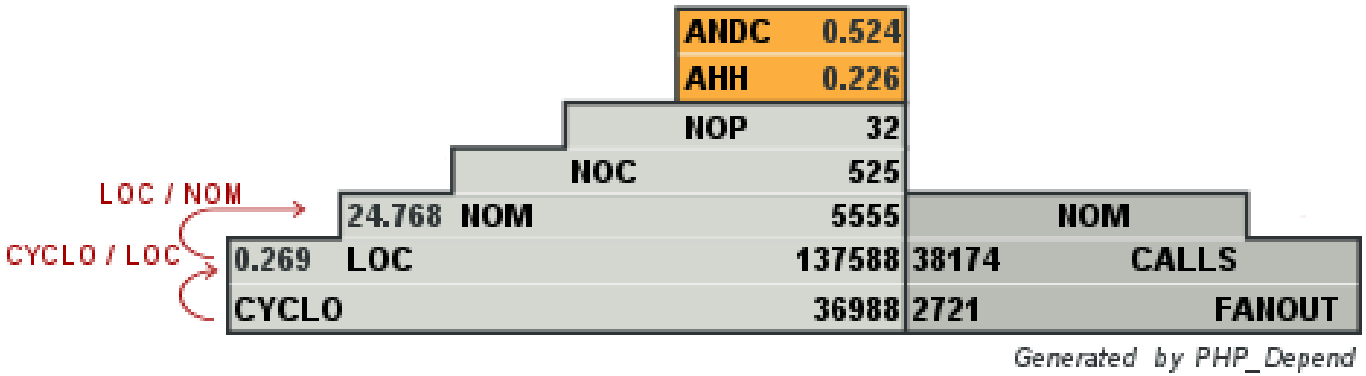
Overview Pyramid

- Beziehungen zwischen den Metriken
  - Zu erwartende Code-Komplexität



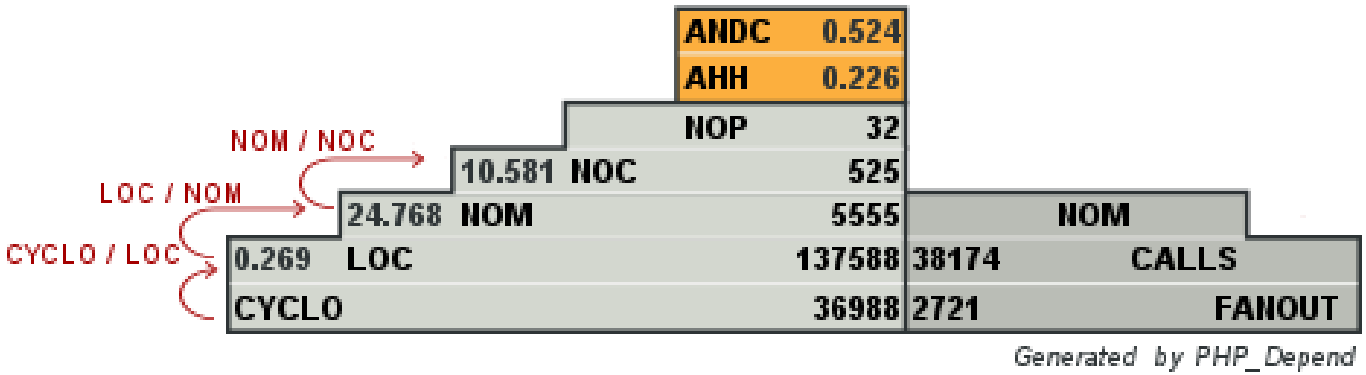
Overview Pyramid

- Beziehungen zwischen den Metriken
  - Durchschnittliches Methoden-Volumen



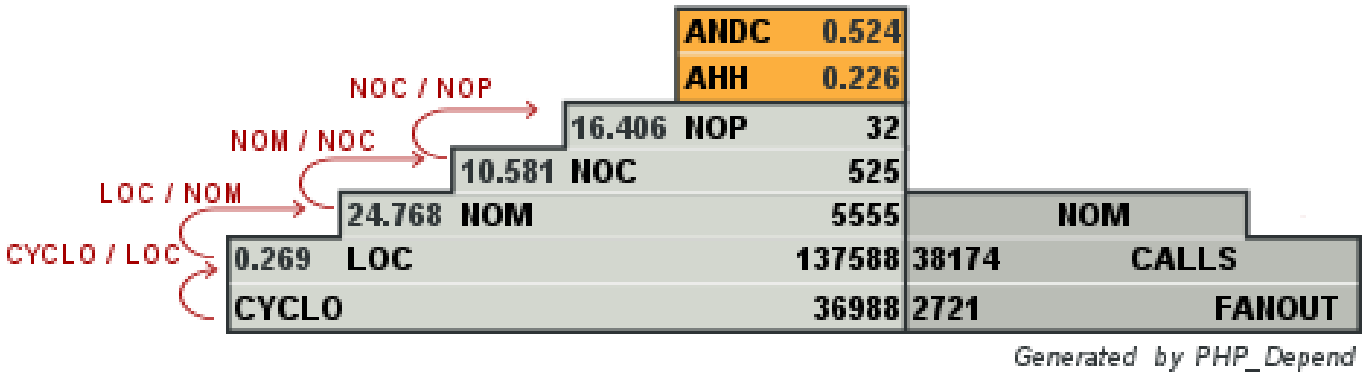
Overview Pyramid

- Beziehungen zwischen den Metriken
  - Indikator für das verwendete Klassendesign



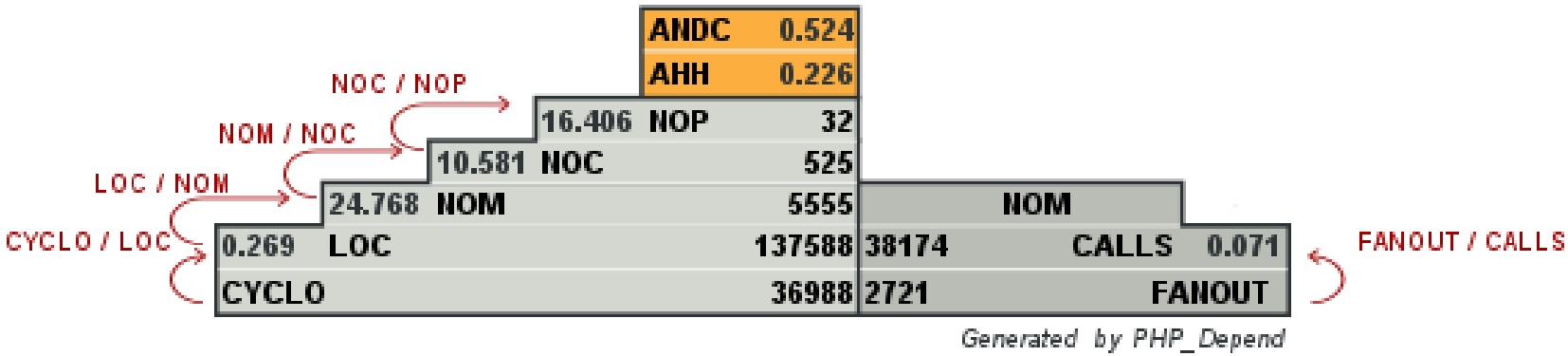
Overview Pyramid

- Beziehungen zwischen den Metriken
  - Globale Projektstruktur



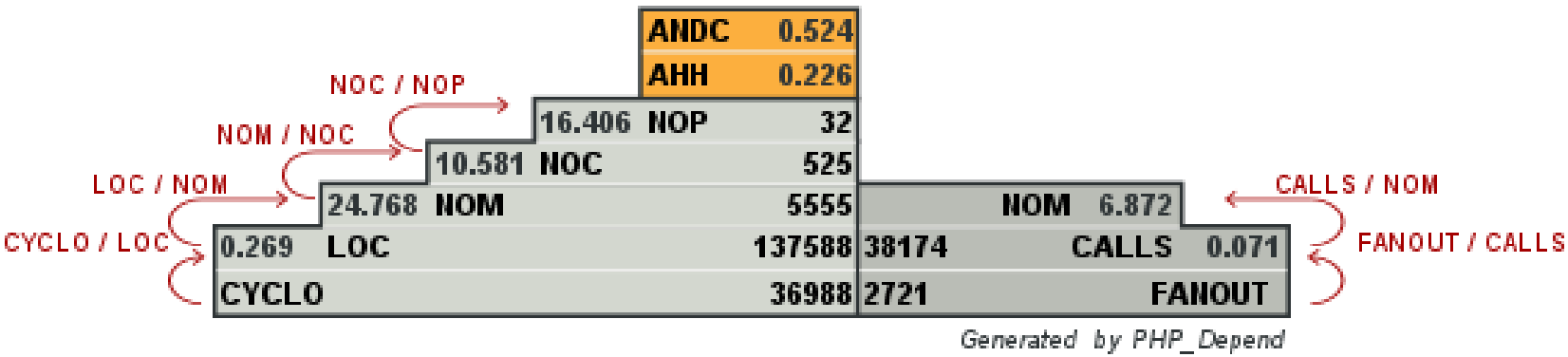
Overview Pyramid

- Beziehungen zwischen den Metriken
  - Verteilung von Logik im System



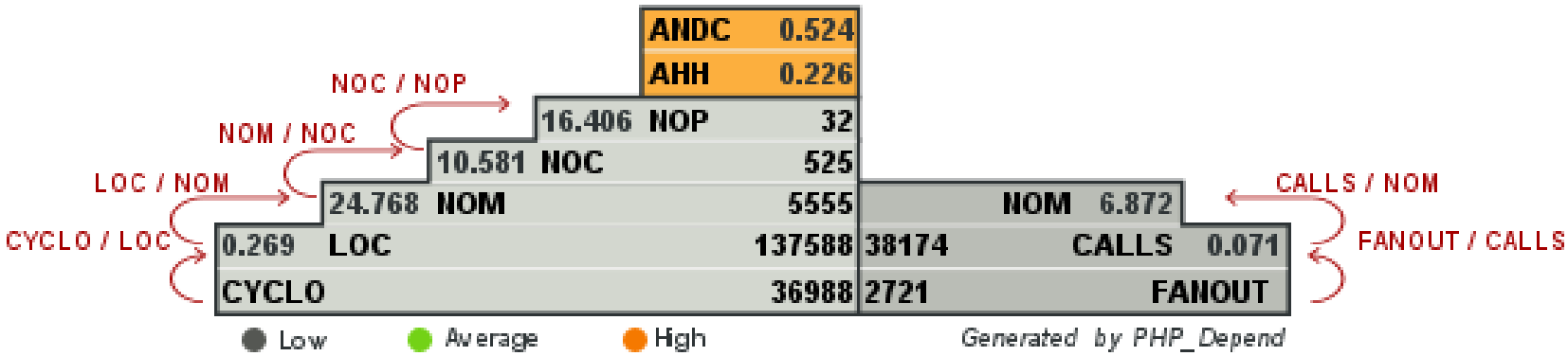
Overview Pyramid

- Beziehungen zwischen den Metriken
  - Kopplung/Kapselung von Daten und Methoden



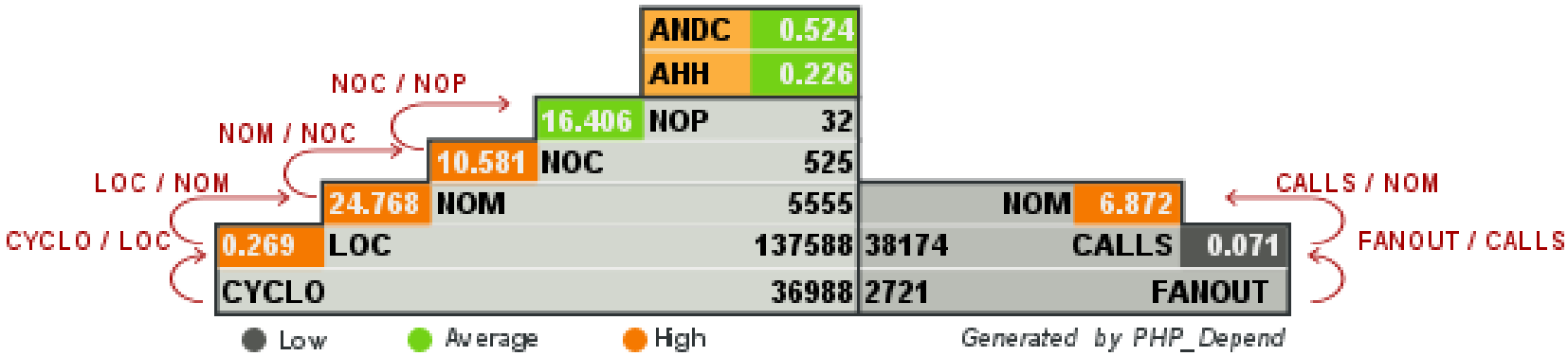
Overview Pyramid

- Schwellenwerte helfen bei der Interpretation



Overview Pyramid

- Schwellenwerte helfen bei der Interpretation
  - Farben mit einer semantischen Bedeutung (Rot/Grün-Effekt)



Outline

Metriken sind . . .

Metriken sind . . .

- . . . keine Magie, sondern einfache Maßzahlen

Metriken sind . . .

- . . . keine Magie, sondern einfache Maßzahlen
- . . . nutzlos, ohne Grenz- oder Referenzwerte

Metriken sind . . .

- . . . keine Magie, sondern einfache Maßzahlen
- . . . nutzlos, ohne Grenz- oder Referenzwerte
- . . . skalierbar, wachsen mit der Projektgröße

Metriken sind . . .

- . . . keine Magie, sondern einfache Maßzahlen
- . . . nutzlos, ohne Grenz- oder Referenzwerte
- . . . skalierbar, wachsen mit der Projektgröße
- . . . automatisierbar und reproduzierbar

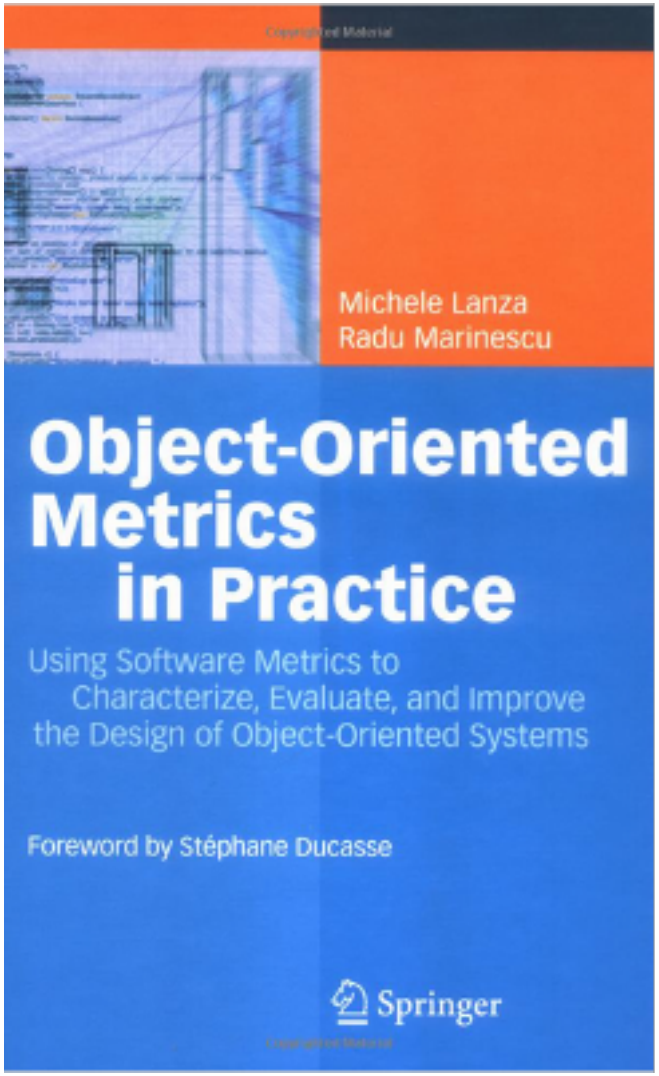
Metriken sind . . .

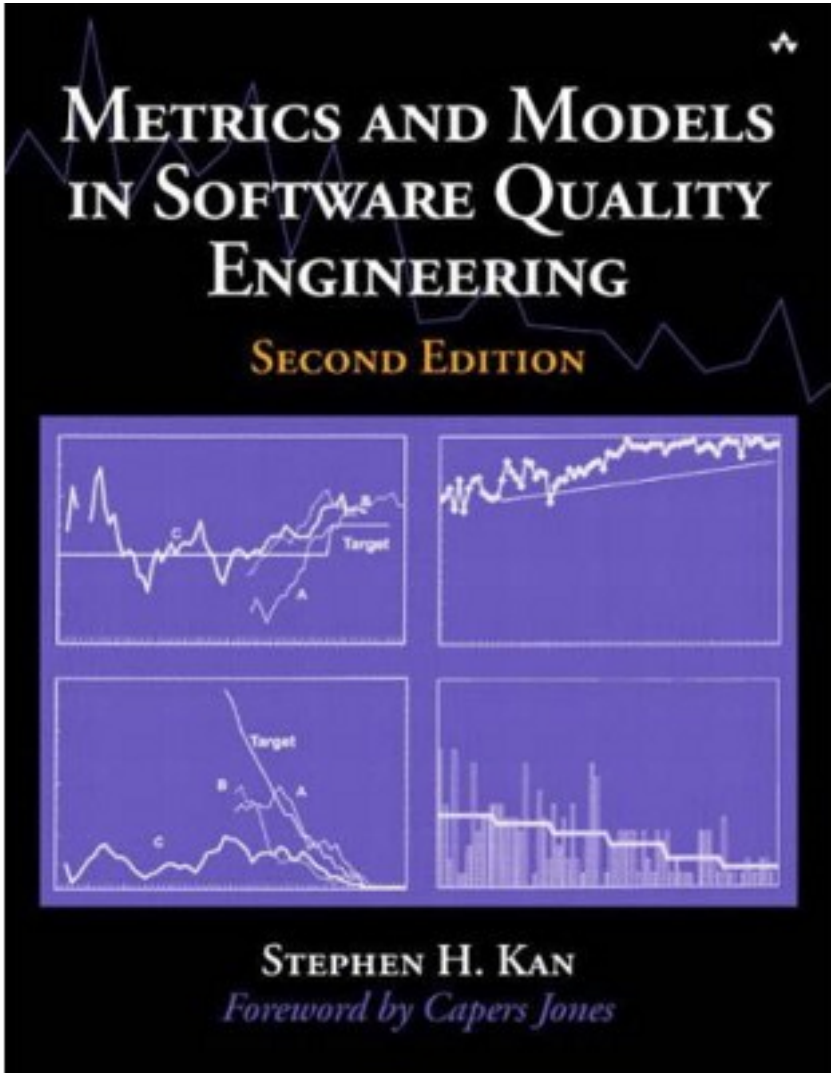
- . . . keine Magie, sondern einfache Maßzahlen
- . . . nutzlos, ohne Grenz- oder Referenzwerte
- . . . skalierbar, wachsen mit der Projektgröße
- . . . automatisierbar und reproduzierbar
- . . . objektiv, da Software gestützt

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

Buchempfehlung





Vielen Dank für Ihre Aufmerksamkeit

Fragen? Kommentare? Kritik? Ideen?

Vielen Dank für Ihre Aufmerksamkeit

Die Folien finden Sie bald unter
`http://talks.qafoo.com`

Vielen Dank für Ihre Aufmerksamkeit

Kontakt

- Manuel Pichler
- manuel@qafoo.com
- @manuelp / @qafoo