



Jan Schumann, G+J

Manuel Pichler, Trainer & Consultant - Qafoo

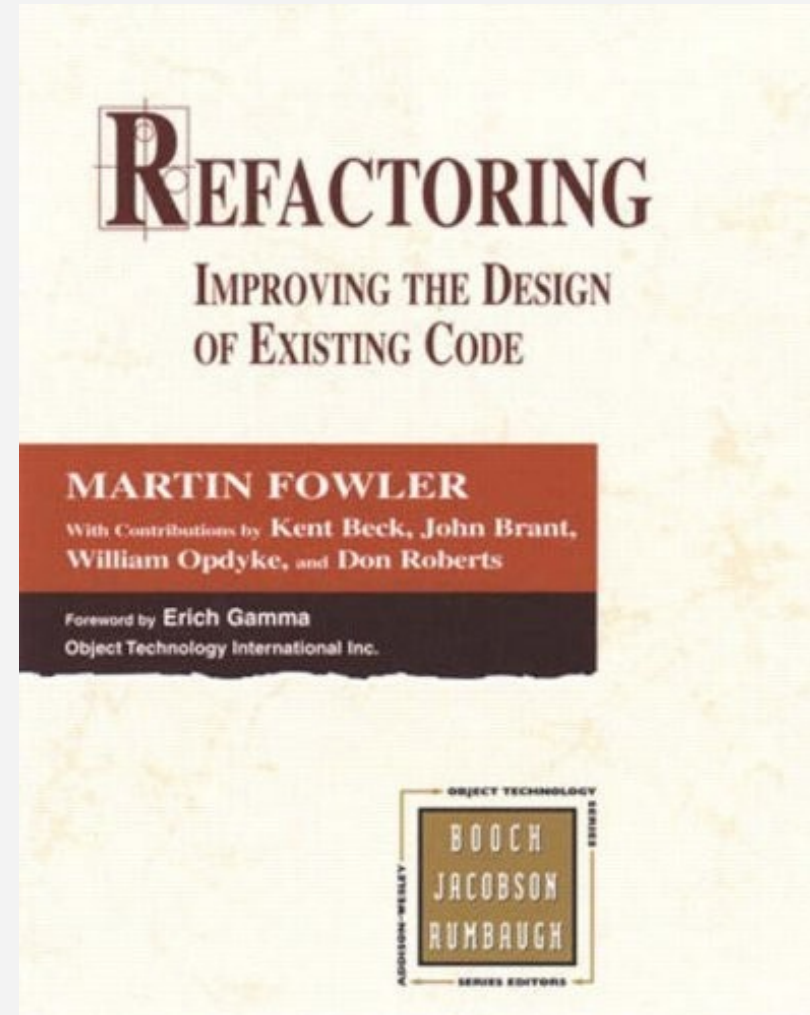
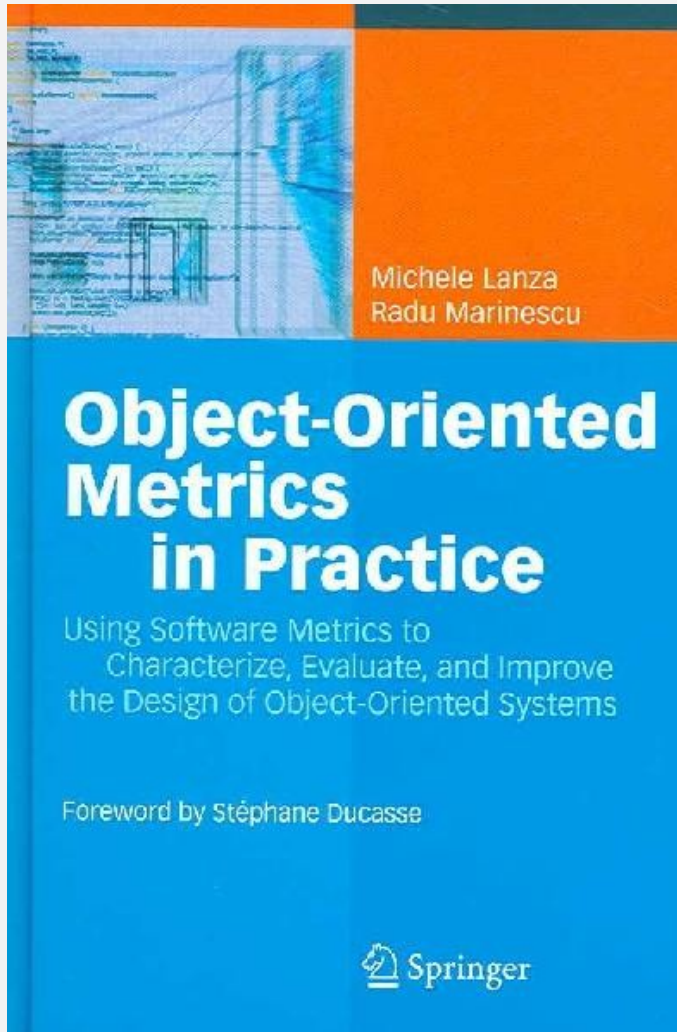
# Statische Codeanalyse wirklich effektiv nutzen



# Über uns

- Jan Schumann
  - Jahrgang 1976
  - System- / Softwarearchitekt
  - Entwickler von:
    - PHP\_Depend, ImageTransform
- Manuel Pichler
  - Jahrgang 1978
  - Softwarearchitekt, Trainer & Consultant - Qafoo
  - Entwickler von:
    - PHP\_Depend, phpUnderControl, PHPMD

# Buchempfehlung



# Was bedeutet Qualität?

*You cannot control what you cannot measure.*

Tom DeMarco

# Festlegen von Qualitätskriterien

- Qualität ist immer relativ, ausgehend vom Betrachter
  - ⊙ Nutzer haben ihre eigene Ansprüche
  - ⊙ Kunden und Betreiber wieder Andere
  - ⊙ Und Entwickler sowieso...

# Mögliche Qualitätskriterien

- Funktionalität (Benutzer)
- Verlässlichkeit (Benutzer, Betreiber)
- Benutzbarkeit (Benutzer)
- Effektivität (Betreiber)
- Wartbarkeit (Kunde, Entwickler)
- Portabilität (Betreiber)

# Widerspruch nicht ausgeschlossen

## ● Porsche vs. Golf

- ⊙ Beide erfüllen die Qualitätsansprüche an ein Auto
  - ⊙ Befördert von Ort A nach B
  - ⊙ Zuverlässigkeit
  - ⊙ Wartbarkeit
- ⊙ Aber mit ganz unterschiedliche Komfortkriterien
  - ⊙ Performance
  - ⊙ Benutzbarkeit

# Konsequenz

- Es gibt keine beste Qualität, sondern nur die passende Qualität
  - ⦿ Durch Auswahl / Kombination geeigneter Qualitätskriterien wird eine statische Analyse erst möglich

# Beispiel Wartbarkeit

- ISO/IEC 9126 definiert Wartbarkeit durch:
  - ⊙ Wiederverwendbarkeit
  - ⊙ Erlernbarkeit
  - ⊙ Analysierbarkeit
  - ⊙ Anpassbarkeit
  - ⊙ Stabilität
  - ⊙ Testbarkeit

# Warum ist Software nicht wartbar?

- Aufgrund zu hoher Komplexität
- Durch falsch genutzte Vererbung
  - ⊙ Wegen Missachtung grundlegender Architekturprinzipien
  - ⊙ Verletzung des Single-Responsibility-Prinzips
  - ⊙ Unzureichendes Information-Hiding
  - ⊙ Liskovsches Substitutionsprinzip

# Symptome

- Bezogen auf einzelne Klassen
  - ⊙ Hohe Komplexität
  - ⊙ Zugriff auf externe Daten
  - ⊙ Geringer Zusammenhalt
- Bezogen auf Klassenhierarchien
  - ⊙ Geringe Wiederverwendung geerbter Methoden
  - ⊙ Geringe Spezialisierung
  - ⊙ Starke Erweiterung der Schnittstelle

# Blick auf das Ganze

- Einzelne Metriken
  - ⊙ Sind ein guter Indikator
  - ⊙ Codezeilen, Komplexität, ...
- Kombination von Metriken
  - ⊙ Geben wesentlich tiefere Einblicke
  - ⊙ Erkennung von Code-Smells
- Schwellenwerte
  - ⊙ Werden zur Einordnung benötigt

# Design Disharmonies / Code Smells

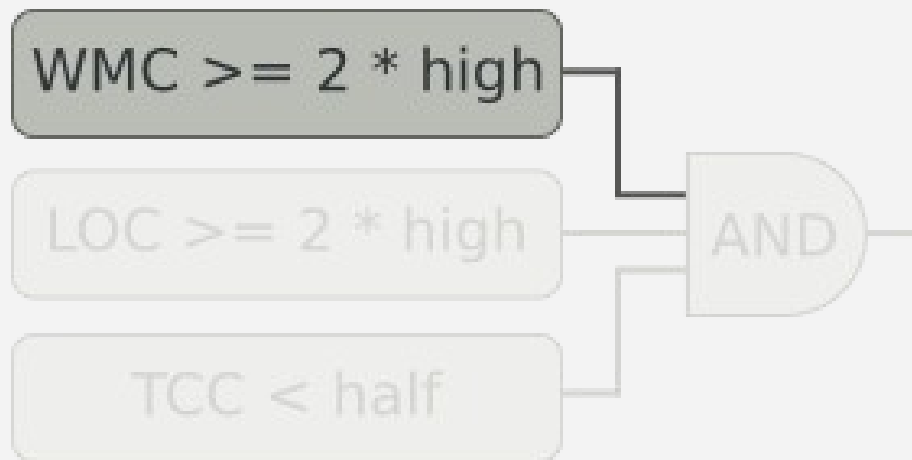
## Identity

- ⊙ Klasse übernimmt mehrere Aufgaben
- ⊙ Unausgewogenes Verhältnis zwischen Daten und Operationen

## Classification

- ⊙ Vererbung dient primär der Wiederverwendung
- ⊙ Die Schnittstelle der Basisklasse wird zu stark erweitert
- ⊙ Die Austauschbarkeit von Implementierung ist nicht gewährleistet

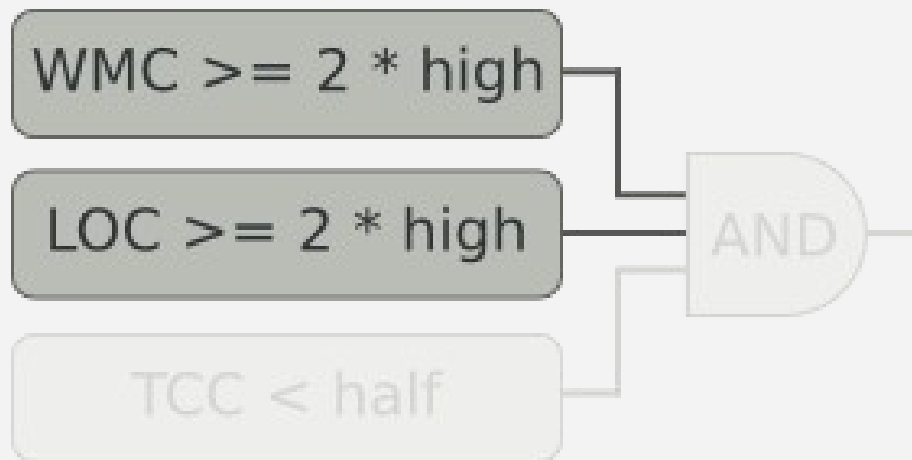
# Identity - Design Disharmony



● WMC

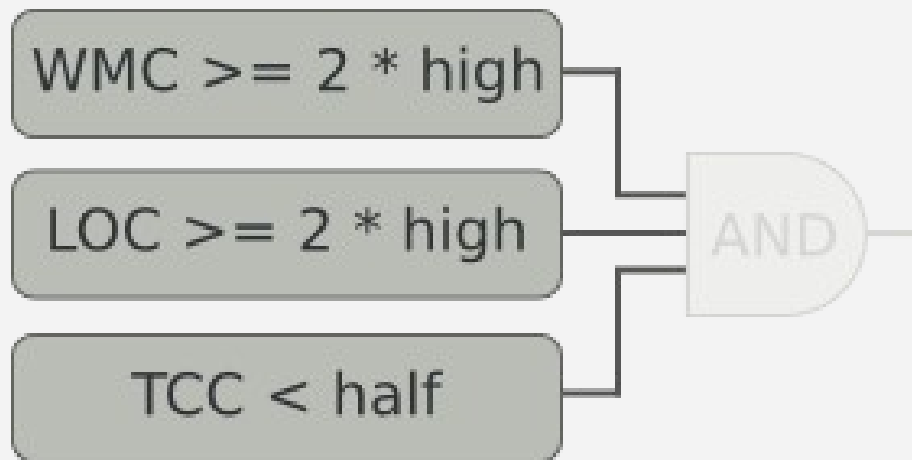
⊙ Weighted Method per Class

# Identity - Design Disharmony



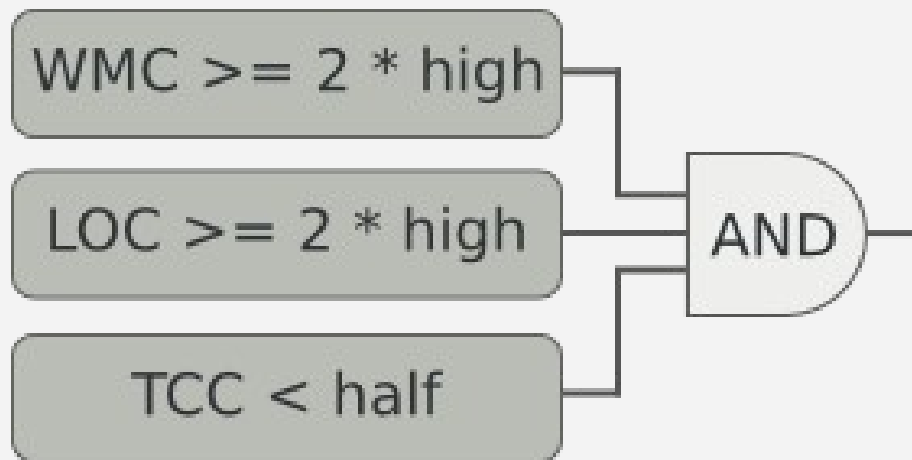
- WMC
  - ⦿ Weighted Method per Class
- LOC
  - ⦿ Lines Of Code

# Identity - Design Disharmony



- WMC
  - ⦿ Weighted Method per Class
- LOC
  - ⦿ Lines Of Code
- TCC
  - ⦿ Tight Class Cohesion

# Identity - Design Disharmony



## ● WMC

- ⊙ Weighted Method per Class

## ● LOC

- ⊙ Lines Of Code

## ● TCC

- ⊙ Tight Class Cohesion

# Identity – Ausprägungen

## ● God Class / Large Class

- ⊙ Erfüllt eine Vielzahl von Aufgaben im System und verwendet meist Daten anderer Klassen

## ● Feature Envy

- ⊙ Implementiert Verhalten, das zumeist auf den Daten anderer Klassen operiert

## ● Data Class

- ⊙ Implementiert selbst kein Verhalten, stellt aber seine Daten direkt oder indirekt bereit

# Identity – Refactorings

## ● God Class

- ⊙ Extract Class (149), Extract Subclass (330)
- ⊙ Extract Interface (341)

## ● Feature Envy

- ⊙ Extract Method (110), Move Method (142)
- ⊙ Move Field (146)

## ● Data Class

- ⊙ Extract Method (110), Move Method (142)
- ⊙ Hide Method (303)

# Identity – Beispiel

## Feature Envy

```
class Calculator {  
  
    protected $o = null;  
  
    // ...  
  
    public function divide() {  
        $r = $this->o->getRight();  
        $l = $this->o->getLeft();  
        if ($r === 0) {  
            return 0;  
        }  
        return $l / $r;  
    }  
}
```

## Data Class

```
class Operands {  
  
    protected $left = 0;  
    protected $right = 0;  
  
    // ...  
  
    public function getLeft() {  
        return $this->left;  
    }  
  
    public function getRight() {  
        return $this->right;  
    }  
}
```

# Identity – Beispiel

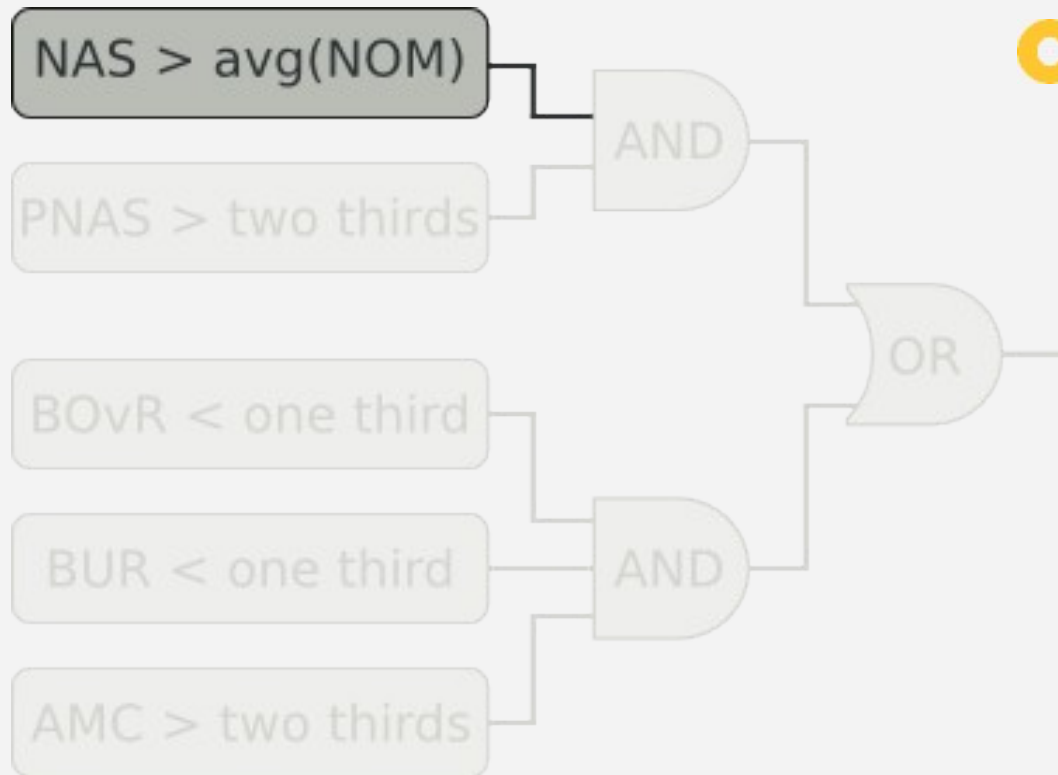
## Move Field (146)

```
class Calculator {  
  
    protected $left = 0;  
    protected $right = 0;  
  
    // ...  
  
    public function divide() {  
        if ($this->right === 0) {  
            return 0;  
        }  
        return $this->left /  
            $this->right;  
    }  
}
```

## Move Method (142)

```
class Operands {  
  
    protected $left = 0;  
    protected $right = 0;  
  
    // ...  
  
    public function divide() {  
        if ($this->right === 0) {  
            return 0;  
        }  
        return $this->left /  
            $this->right;  
    }  
}
```

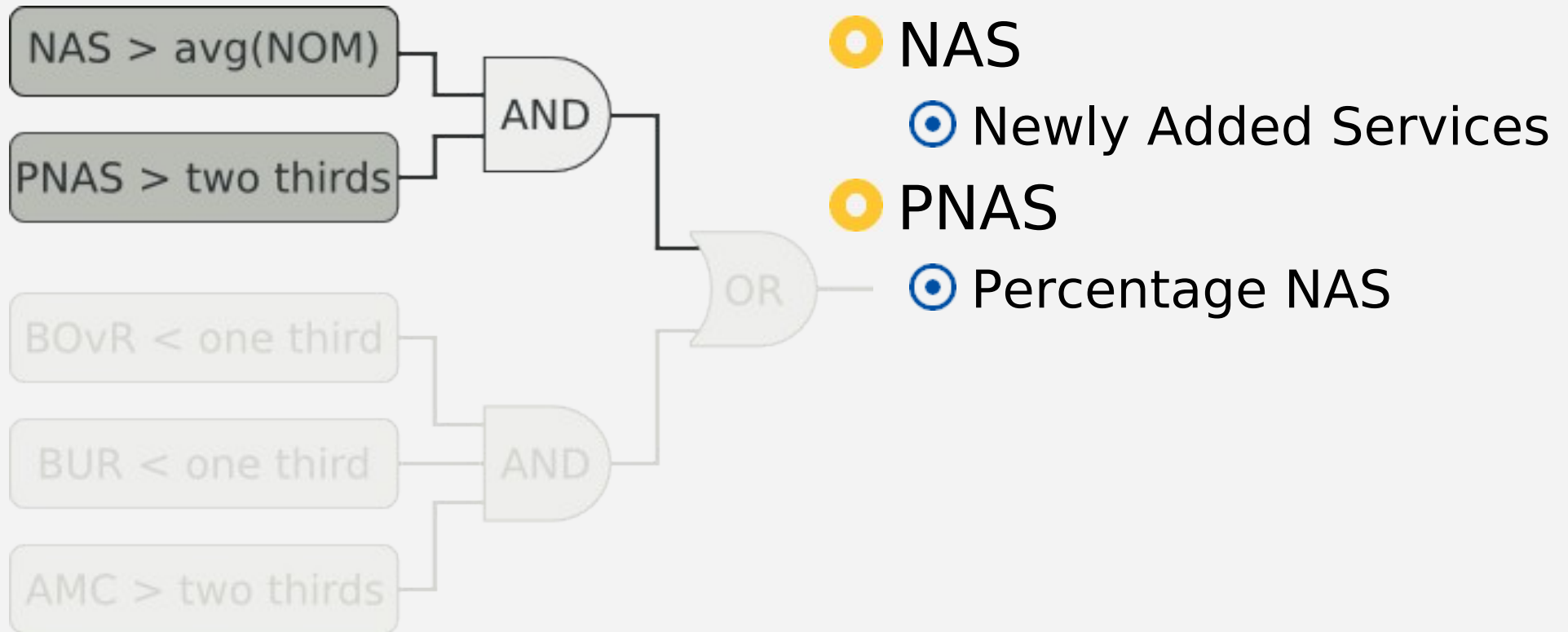
# Classification - Design Disharmony



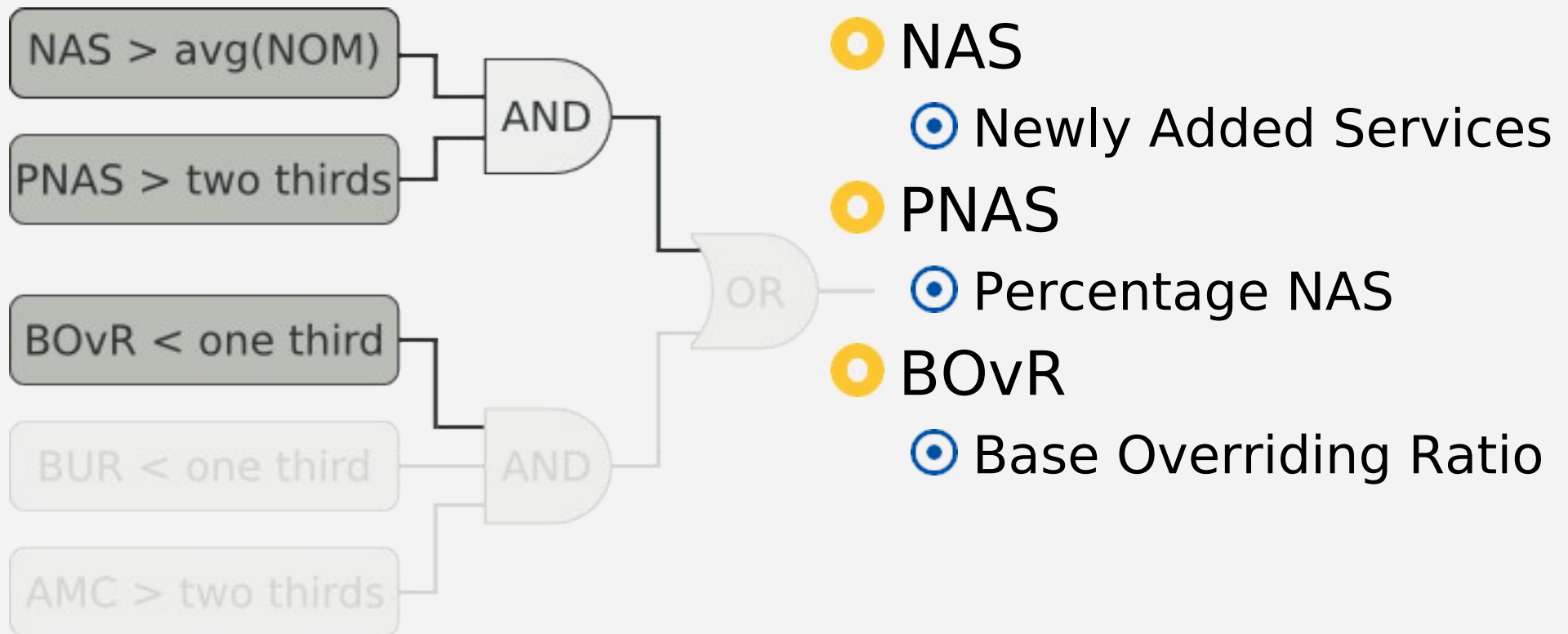
● NAS

⦿ Newly Added Services

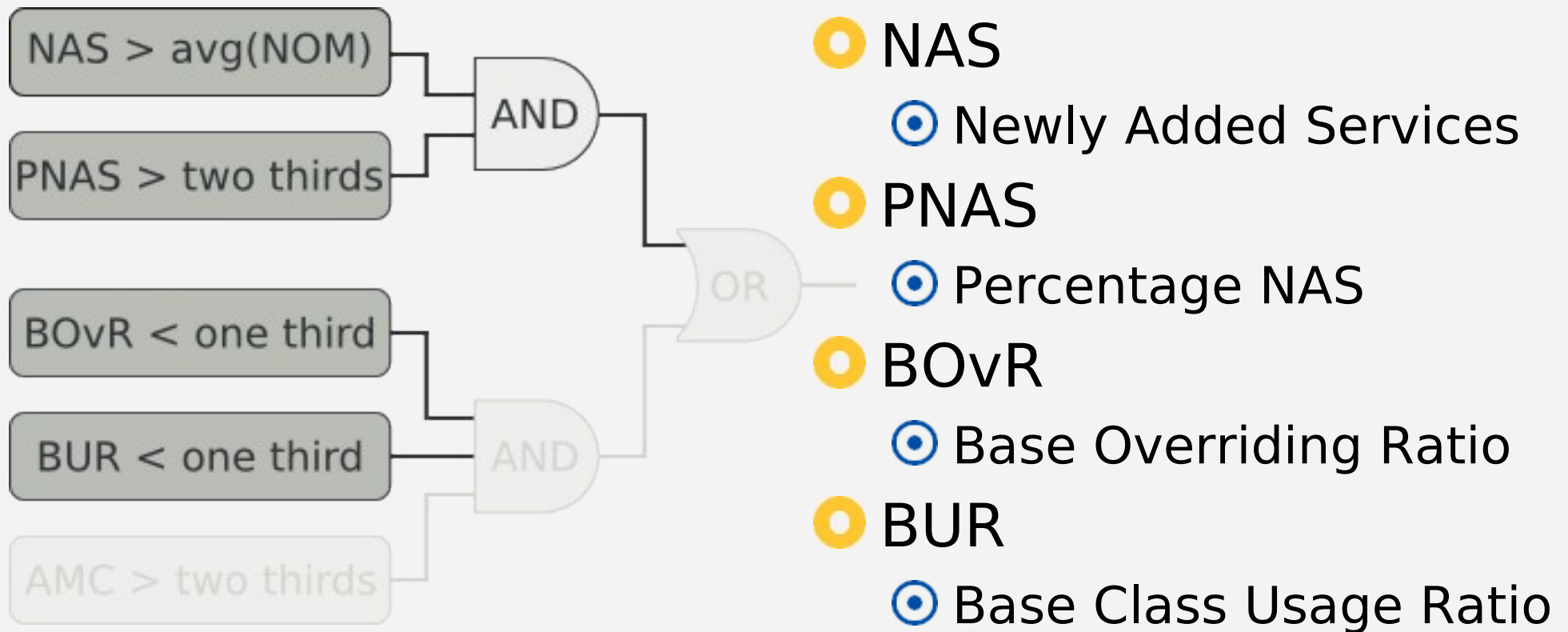
# Classification - Design Disharmony



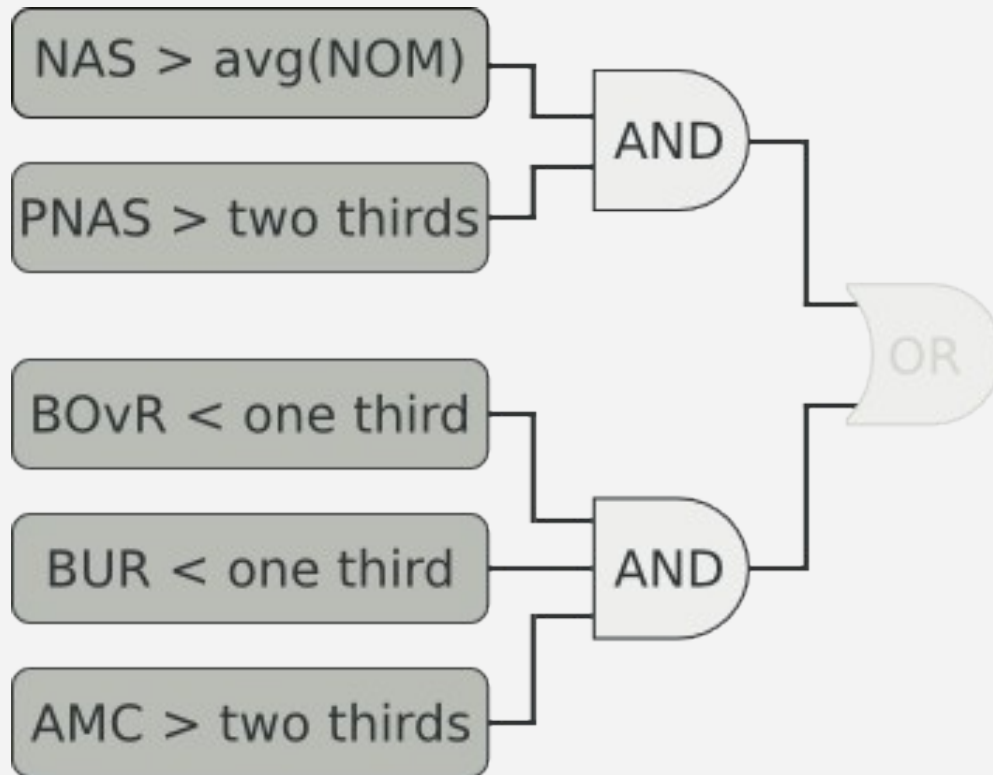
# Classification - Design Disharmony



# Classification - Design Disharmony

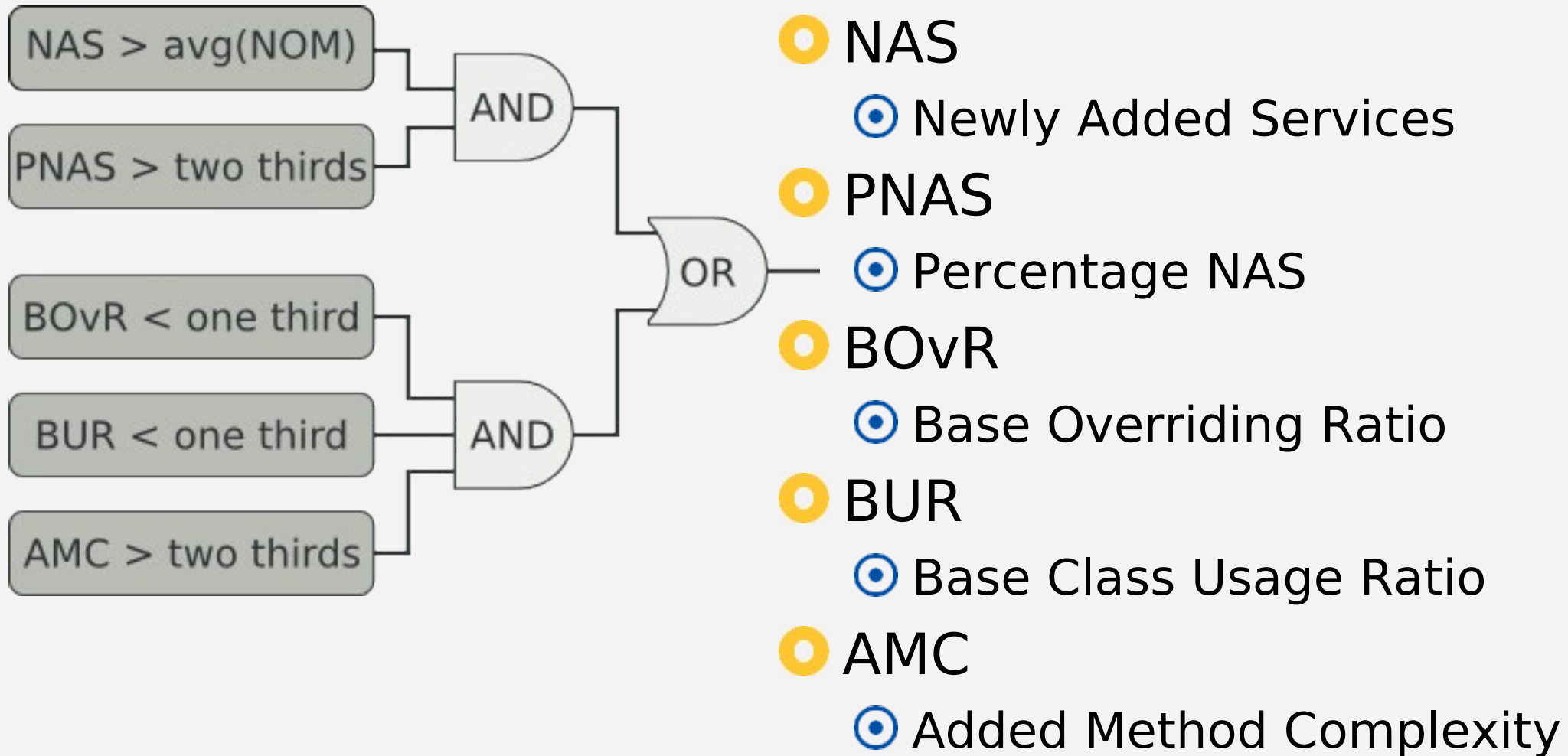


# Classification - Design Disharmony



- NAS
  - ⊙ Newly Added Services
- PNAS
  - ⊙ Percentage NAS
- BOvR
  - ⊙ Base Overriding Ratio
- BUR
  - ⊙ Base Class Usage Ratio
- AMC
  - ⊙ Added Method Complexity

# Classification - Design Disharmony



# Classification - Ausprägungen

## ● Refused Parent Bequest

- ⊙ Ignoriert die Elternklasse, implementiert viele Dienste neu (Code-Duplizierung)
- ⊙ Die Elternklasse bietet Dienste, die nur eine Untermengen der abgeleiteten Klassen nutzt

## ● Tradition Breaker

- ⊙ Ignoriert die Schnittstelle der Elternklasse
- ⊙ Führt eine Vielzahl neuer Dienste ein, die eine Substitution unterbinden

# Classification - Refactorings

## ● Refused Parent Bequest

- ⊙ Replace Inheritance with Delegation (352)
- ⊙ Push Down Method (328)
- ⊙ Push Down Field (329)

## ● Tradition Breaker

- ⊙ Hide Method (303)
- ⊙ Pull Up Method (322)
- ⊙ Extract Class (149), Extract Subclass (330)

# Classification - Beispiel

## ○ Refused Parent Bequest

```
abstract class List {
    public abstract function getAll();
    protected function filter($objects) { /* ... */ }
}

class DocumentList extends List {
    public function getAll() { return $this->filter($this->docs); }
}

class ImageList extends List {
    public function getAll() { return $this->filter($this->images); }
}

class CommentList extends List {
    public function getAll() { return $this->comments; }
}
```

# Classification - Beispiel

## 🟡 Replace Inheritance with Delegation (352)

```
abstract class List {
    protected $helper = null;
    public abstract function getAll();
}

class DocumentList extends List {
    public function getAll() { return $this->helper->filter($th..); }
}

class ImageList extends List {
    public function getAll() { return $this->helper->filter($th..); }
}

class CommentList extends List { /* ... */ }

class FilterHelper implements Filter {
    public function filter(List $this) { /* ... */ }
}

31
```

# Schwellenwerte

- Ein interessanter, aber auch umstrittener Aspekt beim Einsatz von Softwaremetriken sind die verwendeten Schwellenwerte
  - ⊙ Denn sie..
    - ⊙ ... liegen immer im Ermessen des Betrachters
    - ⊙ ... haben großen Einfluss auf das Endergebnis
  - ⊙ Ein guter Ausgangspunkt für die Bestimmung von Schwellenwerten sind gemessene Mittelwerte ...
    - ⊙ ... eigener Projekte
    - ⊙ ... von Open-Source-Projekten

# Danke

*However, a metric is not a god, it is merely  
a measurement against an arbitrary standard*

Robert C. Martin

# Verwendete Softwaremetriken

- CYCLO: Zyklomatische Komplexität [1-n]
- WMC: Weighted Method per Class [0-n]
- TCC: Tight Class Cohesion [0-1]
- LOC: Lines Of Code [1-n]
- BUR: Base Class Usage Ratio [0-1]
- BOvR: Base Class Overriding Ration [0-1]
- NAS: Newly Added Service [0-n]
- PNAS: Percentage of NAS [0-1]