Metriken in der Softwareentwicklung

von

Markus Weßjohann

Alte Oldenburger AG

Bekannte Metriken

- Lines of Code
- Code Coverage
- McCabe-Metrik
- Anzahl Fehler

Welches Programm ist einfacher?

Programm 0

```
XXXXXXXXXXXXXXX
   XXXXX
   XXXXXXXXXX
      XXXXXXXXXXX
         XXXXX
      X
   X
   XXXXXXXXXX
      XXXXXXXXXXX
         XXXXX
      X
   X
   XXXXXXXXXX
      XXXXXXXXXXXXX
         XXXXX
      X
   X
X
```

Programm 1

```
XXXXXXXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXXXXXXXX
      XXXXX
   X
   XXXXX
   XXXXX
   XXXXX
   XXXXX
X
```

Gedanken-übertragung

~75%

Programm 1

Programm 0

```
XXXXXXXXXXXXXXX
   XXXXX
   XXXXXXXXXX
      XXXXXXXXXXX
         XXXXX
      X
   X
   XXXXXXXXXX
      XXXXXXXXXXX
         XXXXX
      X
   X
   XXXXXXXXXX
      XXXXXXXXXXXXX
         XXXXX
      X
   X
X
```

Programm 1

```
XXXXXXXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXXXXXXXXX
      XXXXX
   X
   XXXXX
   XXXXX
   XXXXX
   XXXXX
X
```

Programm 0

```
XXXXXXXXXXXXXXX
   XXXXX
   XXXXXXXXXX
      XXXXXXXXXXX
         XXXXX
      X
   X
   XXXXXXXXXX
      XXXXXXXXXXX
         XXXXX
      X
   X
   XXXXXXXXXX
      XXXXXXXXXXXXX
         XXXXX
      X
   X
X
```

Programm 1

```
XXXXXXXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXX
   XXXXXXXXXXXX
      XXXXX
   X
   XXXXX
   XXXXX
   XXXXX
   XXXXX
X
```

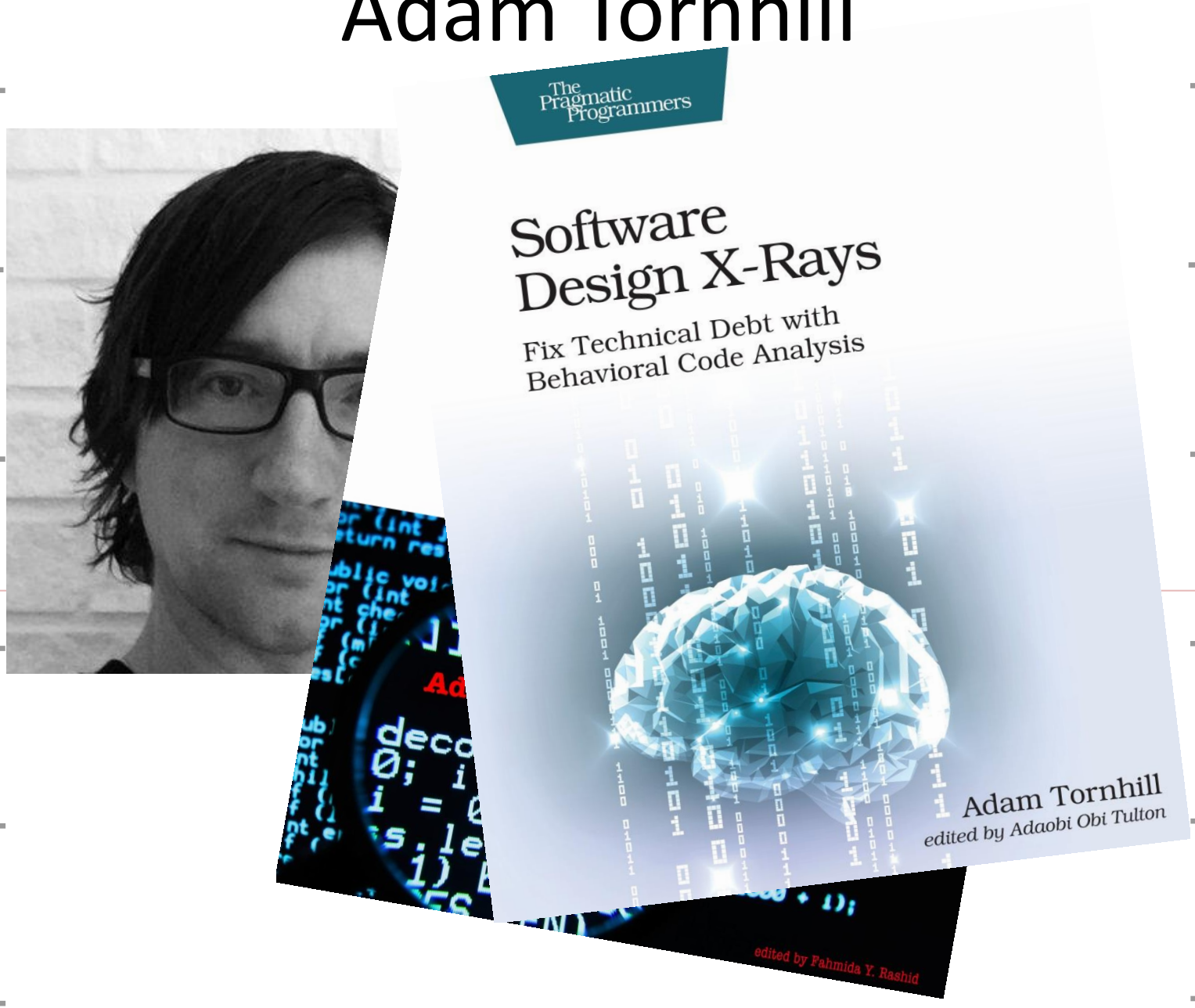
Programm 0

```
xxxxxxxxxxxxxxx
@@xxxxx
@@xxxxxxxxxx
@@@@xxxxxxxxxxx
@@@@@@xxxxx
@@@@x
@@x
@@xxxxxxxxxx
@@@@xxxxxxxxxxx
@@@@@@xxxxx
@@@@x
@@x
@@xxxxxxxxxx
@@@@xxxxxxxxxxx
@@@@@@xxxxx
@@@@x
@@x
x
```

Programm 1

```
xxxxxxxxxx
@@xxxxx
@@xxxxx
@@xxxxx
@@xxxxx
@@xxxxx
@@xxxxx
@@xxxxx
@@xxxxx
@@xxxxxxxxxxx
@@@@xxxxx
@@x
@@xxxxx
@@xxxxx
@@xxxxx
@@xxxxx
x
```

Adam Tornhill



Was wird benötigt?

- Python
- https://github.com/adamtornhill/maat-scripts

- git - optional

Auswerten der Ergebnisse

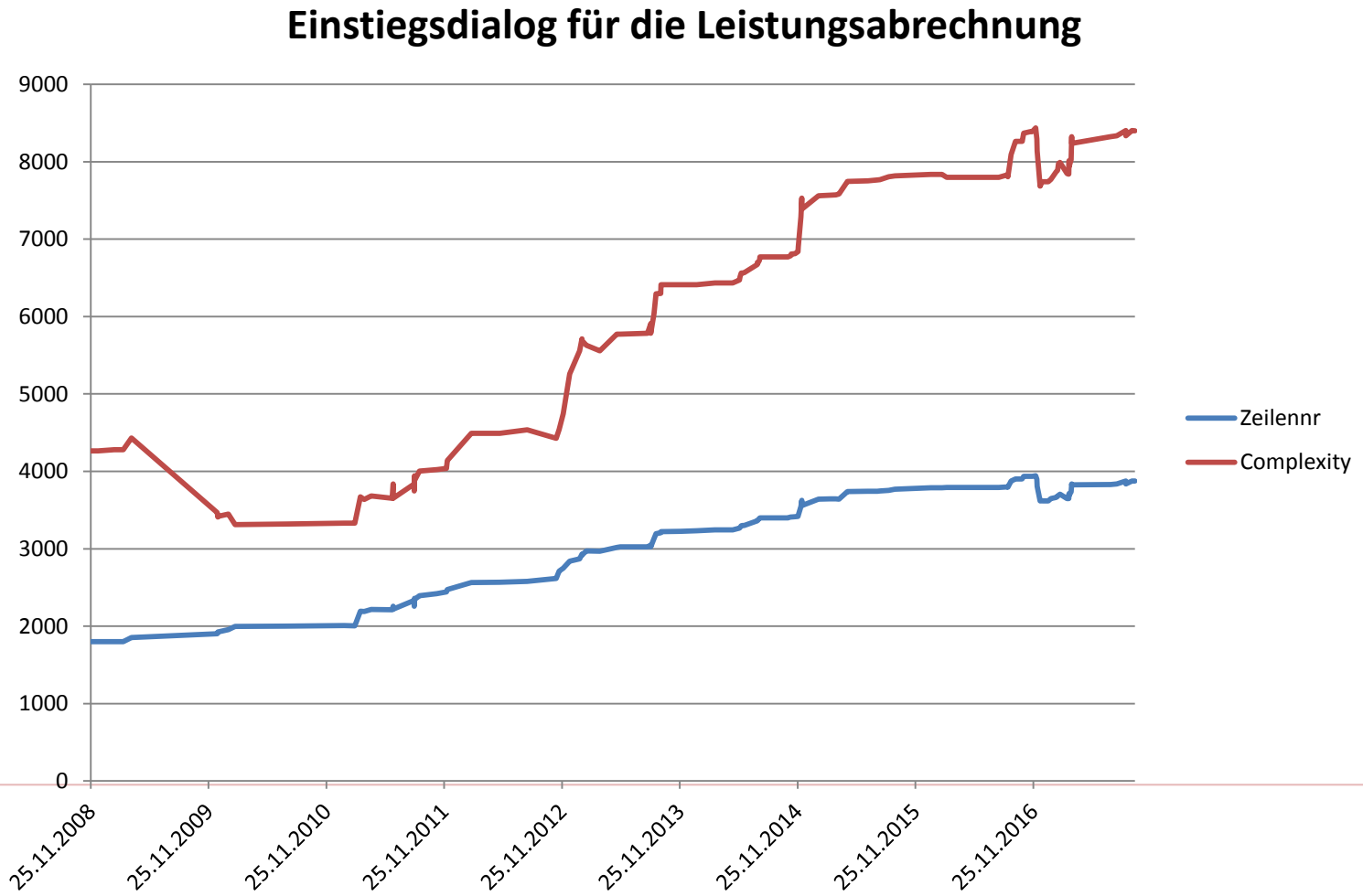
- Ergebnisse werden beeinflusst von:
  - Programmiersprache
  - Programmierrichtlinien
  - Programmierstil
- Kein Richtwert von mir
  ,aber
- Richtwert ~ 90% Dateien

Start der Analyse

```
~> complexity_analysis.py VERSIS |
sort -nr -k3 -t';'|
head -5

Modul 1 ; 3463 ;  9517 ; 6,11 ; 4,30 ; 13 …
Modul 2 ; 2073 ;  8230 ; 6,09 ; 1,41 ; 9  …
Modul 3 ; 1346 ;  5411 ; 6,57 ; 2,13 ; 9  …
Modul 4 ; 2230 ;  2665 ; 7,48 ; 1,74 ; 10 …
Modul 5 ; 1209 ;  1582 ; 2,58 ; 3,51 ; 17 …
```

Beispiel


Einstiegsdialog für die Leistungsabrechnung
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
25.11.2008
25.11.2009
25.11.2010
25.11.2011
25.11.2012
25.11.2013
25.11.2014
25.11.2015
25.11.2016
Zeilennr
Complexity

Analyseergebnisse

Mehrere Möglichkeiten:

- Einbinden in git als git-hook
- Einbindung in die IDE/SonarQube/CI
- Manuelle Reports

Vorgehen

Ausprobieren &
Ergebnisse sichern

1. Abbau technischer Schulden
   durch Refactorings
2. Aufwandschätzungen

Zusammenfassung

1. Analysieren
2. Probleme identifizieren
3. Probleme bewerten
4. Probleme abmildern
5. REPEAT

Blogeintrag: Mit Metriken managen: Mist!

Fragen

???

Weiter Informationen

- http://www.adamtornhill.com/
- https://empear.com/products/codescene-on-premise/
- https://github.com/adamtornhill
  - https://github.com/adamtornhill/maat-scripts
  - https://github.com/adamtornhill/code-maat
- https://blog.sonarsource.com/cognitive-complexity-because-testability-understandability

Weiter Informationen

- https://www.heise.de/developer/artikel/Mit-Metriken-managen-Mist-3568010.html