

h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Anti-Pattern

Präsentation zum Seminar Wissenschaftliches Arbeiten in
der Informatik 1
Sommersemester 2016

Christopher Hohberg

Kevin Rojczyk

Agenda

- Einleitung
- Geschichtliche Entstehung
- Abgrenzung zu Design-Pattern & “technische Schuld”
- Anti-Pattern aus 5 verschiedenen Kategorien
- Hilfswerkzeuge zum Erkennen von Anti-Pattern

Anti-Pattern

- **Lösung** eines Problems auf die **falsche Weise**
- Begriff aus der Softwareentwicklung
- Hat sich auf **andere Bereiche ausgeweitet**
- Oft Uneinigkeit, zu welchem Bereich ein Anti-Pattern gehört
- Viele **verschiedene Namen** für gleiche Anti-Pattern
- Wir verwenden nur geläufigste, **englische Bezeichnungen**

Geschichtliche Entstehung

- 1974 Grundlegender Gedanke** erstmals von Frederick Brooks in “The Mythical Man-Month”
- 1994** Buch über Design-Pattern von “Gang of four” löst **wachsendes Bewusstsein** für Pattern aus.
- 1995** Frederick Brooks Idee wird in verschiedensten Arbeiten **wieder aufgegriffen**, aber nie vertieft.
- 1996** Michael Akroyd nennt **erstmalig konkret** den Begriff “Anti-Pattern” in seinem Vortrag auf der Object World West Conference.

Abgrenzung zu Design-Pattern & “technische Schuld”

Design-Pattern

Ist wünschenswert

Bietet Grundgerüst einer guten Lösung eines bestimmten Problems

“technische Schuld”

Werden bewusst implementiert

Folgen sind bekannt, bevor schlechter Code geschrieben wird.

Anti-Pattern

Sollte vermieden werden

Stellt schlechte Lösung eines bestimmten Problems dar, weil neue Probleme verursacht werden

Anti-Pattern

Werden unbewusst implementiert

Resultierende Probleme kommen überraschend und unerwartet.

Anti-Pattern im Projekt-Management

- Oft ohne IT Bezug
- Große Menge an Literatur speziell zu diesem Thema

Anti-Pattern:

Train the Trainer - die gesamte Gruppe wird fortgebildet, statt einzelne Personen

Hidden Requirement - Projektanforderung ungenau bzw. nicht spezifiziert

Beispiel für Hidden Requirements



How the customer explained it



How the project leader understood it



How the analyst designed it



How the programmer wrote it



What marketing advertised



What the customer really needed

Bild von projectcartoon.com,
Creative Commons License
v3

Anti-Pattern in der Software-Architektur

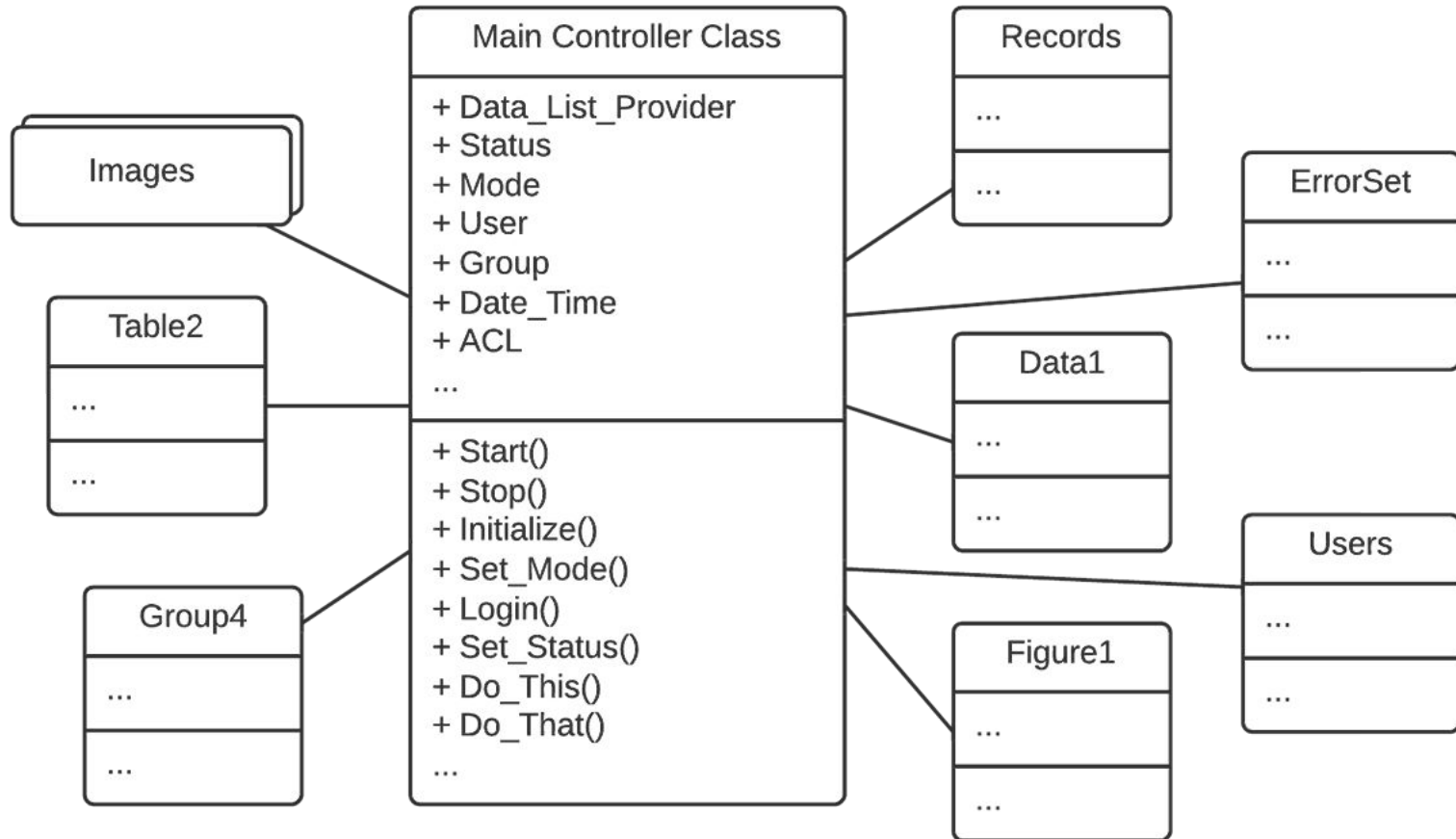
- Fokus liegt auf System Ebene von Anwendungen
- Beschreiben Fehler im Design von Objekten und Klassen und deren Interaktionen

Anti-Pattern:

Swiss Army Knife - Einzelne Klasse versucht Funktionalität für zu viele andere Klassen bereitzustellen.

The Blob - Einzelne Klasse besitzt fast sämtliche Funktionalität eines ganzen Systems.

Klassendiagramm eines Blob



<https://sourcemaking.com/antipatterns/the-blob>

Anti-Pattern in der Softwareentwicklung

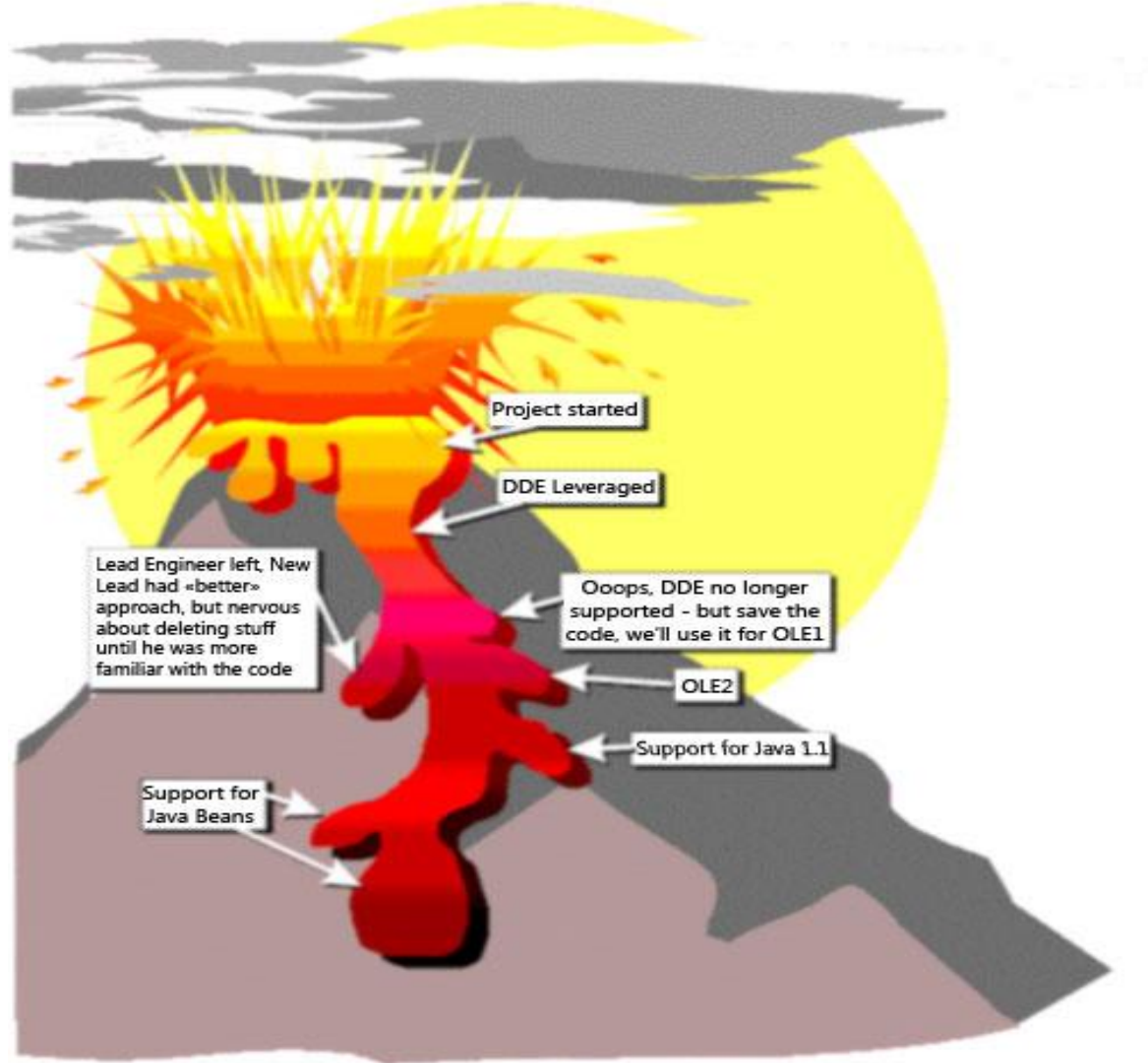
- Probleme in Code Implementierung und Refactoring
- Code umwandeln ohne Funktionalität zu ändern

Anti-Pattern:

Golden Hammer - Allzweckwaffe in der Implementierung. Wird unpassend angewendet.

Lava Flow - Codeblock ohne dokumentierte Funktion und Verwendung.

Beispielgrafik zu Lava Flow



<https://sourcemaking.com/antipatterns/lava-flow>

Anti-Pattern in der testgetriebenen Entwicklung

- Test Driven Development (TDD) ist ein wichtiger Bestandteil der agilen Softwareentwicklung
- Anti-Pattern in TDD haben in der etablierten Literatur keine Bedeutung
- Zusammenstellung aus Communities wie StackOverflow, yahoogroups

Anti-Pattern:

Second Class Citizens - Testcode wird vernachlässigt (Inleserlich, Duplikate)

The Enumerator - test1, test2, test3, usw.

The Free Ride - neue Assertion, statt neue Methode

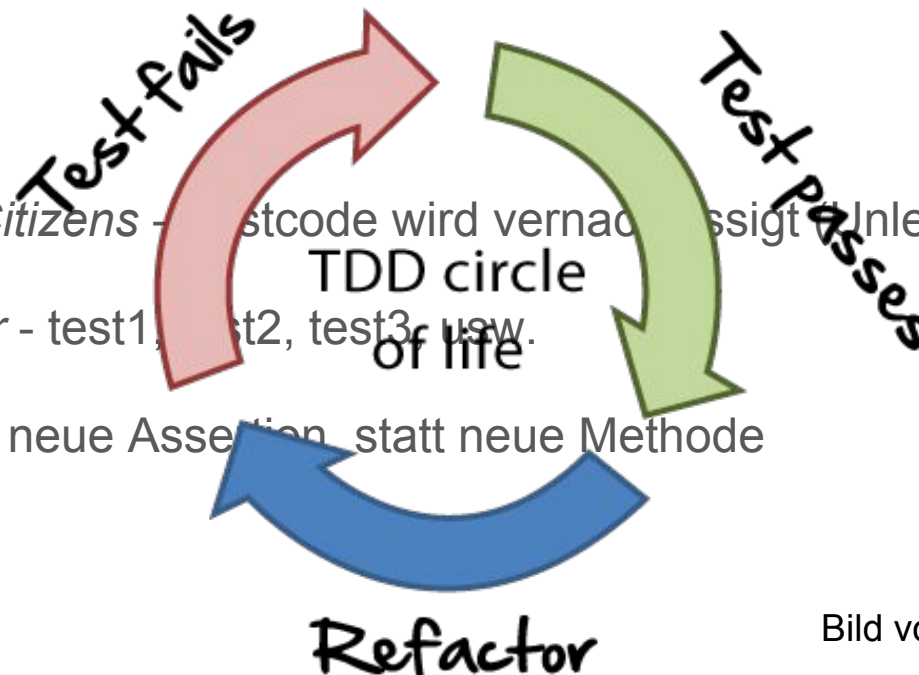


Bild von <http://www.agilenutshell.com>

Anti-Pattern in spezifischen Entwicklungsumgebungen



```
String s = "";  
for (Person p : persons) {  
    s += "," + p.getName();  
}
```

Anti-Pattern für
Programmiersprachen, Bibliotheken,
Betriebssystem, Hardware-Plattformen
...

Übertragbarkeit hängt von der
technischen Umsetzung ab

```
StringBuilder sb = new StringBuilder((persons.size() + 1) * 16);  
for (Person p : persons) {  
    sb.append(p.getName());  
    sb.append(",");  
}
```

Hilfswerkzeuge zum Erkennen von Anti-Pattern

-  **CODE CLIMATE** .com
 - Große Auswahl an Linters
 - Auch lokal nutzbar
-  **quantifiedcode** .com
 - Eigene Pattern
 - Lösungsansätze
 - Berliner Start-Up

Gemeinsame Eigenschaften

- Automatische Überprüfung bei GitHub Commits
- Cloud Dienste
- Frühzeitiges erkennen von Anti-Pattern
- Einheitlicher Code-Style
- Kostenlos für OS Projekte



Das Urheberrecht an den Logos liegt bei den jeweiligen Unternehmen

*“Ich kann freilich nicht sagen, ob es besser werden wird,
wenn es anders wird;
aber so viel kann ich sagen: es muss anders werden,
wenn es gut werden soll.”*

Georg Christoph Lichtenberg, 1853

Vielen Dank für Ihre Aufmerksamkeit

Geschichtliche Entstehung

Frederick Brooks, The Mythical Man-Month, Addison-Wesley, 1975

Erich Gamma, Richard Helm, Ralph E. Johnson, John Vlissides. "Design Patterns. Elements of Reusable Object-Oriented Software", Addison-Wesley, 1994

Michael Akroyd. "AntiPatterns: Vaccinations against Object Misuse" In: San-Jose, Object World West Conference, Aug. 1996

Anti-Pattern in Softwareentwicklung und Softwarearchitektur

<https://sourcemaking.com/>

Anti-Pattern im Projekt Management

William J. Brown, Anti Patterns- Refactoring Software, Architectures, and Projects in Crisis. Wiley, New York, 1998

Anti-Pattern in spezifischen Entwicklungsumgebungen

<http://odi.ch/prog/design/newbies.php>

Anti-Pattern in TDD

<http://blog.james-carr.org/2006/11/03/tdd-anti-patterns/>

<https://stackoverflow.com/questions/333682/unit-testing-anti-patterns-catalogue>