andrena OBJECTS

Experts in agile software engineering

Reverse Engineering Legacy Code

Stefan Mandel

Legacy Code

Typische Anzeichen

- Zu wenige (automatisierte) Tests
- Unkontrollierte Seiteneffekte
  - Auf globale Variablen
  - Und Argumente
- Verzicht auf Objektorientierung
  - Wenig Struktur (Patterns)
  - Generische Typen (Maps)
  - Objekte kapseln nur Daten, nicht Funktionen
- Unbehandelte Fehler

Reverse Engineering

Code verstehen

- Debugging
  - Flexibel aber
  - Nicht automatisierbar
  - Keine Ableitung von reproduzierbaren Testfällen
- Logging
  - Unscharf
  - Information über produktive Nutzung
  - Testfälle unter Umständen ableitbar
- Ziel
  - Genaue Information
  - Auch während der produktiven Nutzung
  - Testfälle müssen ableitbar sein

Charakterisierung

Durch Ableiten von Tests aus laufendem Code

- Aufzeichnen:
  - Fange jeden Aufruf der Methode im realen Betrieb ab
  - Vor der Methode: Serialisiere erhaltene Eingabe-Daten
  - Nach der Methode: Serialisiere erhaltene Ergebnis-Daten
- Testgenerierung:
  - Generiere Testcode aus (Eingabe, Methode, Ergebnis) nach dem **AAA**-Pattern:
    - **Arrange**: Erzeuge alle Eingabe-Variablen aus den serialisierten Eingabewerten
    - **Act**: Rufe Methode mit den Variablen aus Arrange auf
    - **Assert**: Vergleiche alle Ergebnis-Variablen aus Act mit den serialisierten Ergebniswerten
- Regressionstest:
  - Führe den Testcode aus

Charakterisierung - Eingabe

Argumente

```
public boolean method(Collection values) {
   …
}
```


Charakterisierung - Eingabe

```
public boolean method(Collection values) {
  if (this.values == null) {
    …
  }
}
```

This

Charakterisierung - Eingabe

```
public boolean method(Collection values) {
    values.addAll(MyObject.instance.values);
}
```

Globale Variablen

Charakterisierung - Eingabe

```
public boolean method(Collection values) {
    System.in.read();
}
```

Input

Charakterisierung - Ausgabe

Ergebnis

```
public boolean method(Collection values) {
  return values.contains("String");
}
```


Charakterisierung - Ausgabe

Argumente

```
public boolean method(Collection values) {
  values.add("String");
}
```


Charakterisierung - Ausgabe

```
public boolean method(Collection values) {
  this.values.addAll(values);
}
```

This

Charakterisierung - Ausgabe

```
public boolean method(Collection values) {
    MyObject.INSTANCE.addAll(values);
}
```

Globale Variablen

Charakterisierung - Ausgabe

```
public boolean method(Collection values) {
    System.out.println(values);
}
```

Output

Charakterisierung - Ausgabe

```
public boolean method(Collection values) {
    throw new NumberFormatException();
}
```

Ausnahmen

Demo

Reverse Engineering & Testrecorder

Code verstehen

- Ziel
  - Genaue Information
  - Auch während der produktiven Nutzung
  - Testfälle müssen ableitbar sein

Zeichnet sämtlichen erreichbaren Zustand auf

Auch während die Anwendung läuft

Generiert Tests

Legacy Code & Testrecorder

Typische Anzeichen

- Zu wenige (automatisierte) Tests
- Unkontrollierte Seiteneffekte
  - Auf globale Variablen
  - Und Argumente
- Verzicht auf Objektorientierung
  - Wenig Struktur (Patterns)
  - Generische Typen (Maps)
  - Objekte kapseln nur Daten
- Unbehandelte Fehler

Generiert Tests

Zeichnet Seiteneffekte auf:

Typisierte oder generische Objekte

Und Exceptions

Stand des Projekts Testrecorder

http://testrecorder.amygdalum.net/

- Dokumentation:
  - Anleitung zur Konfiguration
  - Anleitung für die Aufzeichnung (mit und ohne Agent)
  - Beispiele
- Funktionsumfang
  - Ausprobieren
  - Komplexe Objekte (z.B. reflect, native, concurrent) funktionieren noch nicht
  - Komplexe Situationen (z.B. recording von overridden Methoden) funktionieren noch nicht
- Tests
  - > 90% Testüberdeckung
  - Bugs werden über automatisierte Tests reproduziert und gefixt
- API ist noch nicht stabil
  - Jede neue Version ist eine Major Version
- Contribution und Feedback willkommen (https://github.com/almondtools/testrecorder)

Vielen Dank!