„TheBlob" Antipattern

A.k.a. God Object/Class, Winnebago

Hintergrund - „The Blob“ der Film

- Science-Fiction Horrorfilm von 1958; Neuverfilmung in 1988
- Handlung:
  - Ein kleiner Meteor schlägt in den nahegelegenen Wald einer Kleinstadt
  - In der Einschlagstelle befand sich eine gallertartige, unförmige Substanz oder Lebensform, welche die Menschen nach und nach angreift, verschlingt und dabei immer größer wird
  - Anfangs glaubt niemand an die Warnungen über den Blob
  - Der Blob scheint nicht aufzuhalten sein und wird weiterhin größer
  - Zum Schluss wird aber seine Schwachstelle (Kälte) entdeckt, worauf er eingefroren und über der Antarktis abgeworfen wird, wo er für jeden unerreichbar bleibt.

Agenda

- ~~Hintergrund~~
- Beschreibung
- Symptome
- Auswirkungen
- Ursachen
- Lösung
- Zusammenfassung / Template

Beschreibung

- Riesige bzw. „allmächtige“ Controller Klasse => The Blob / God Class
- Weiß zu viel oder tut zu viel
- Umgeben von einfachen Daten-Objekt-Klassen
- Klasse deckt sehr großen Teil des Prozesses (oder sogar den gesamten Prozess) ab
- Ähnelt eher einem Prozeduralen Design als Objekt Orientierung

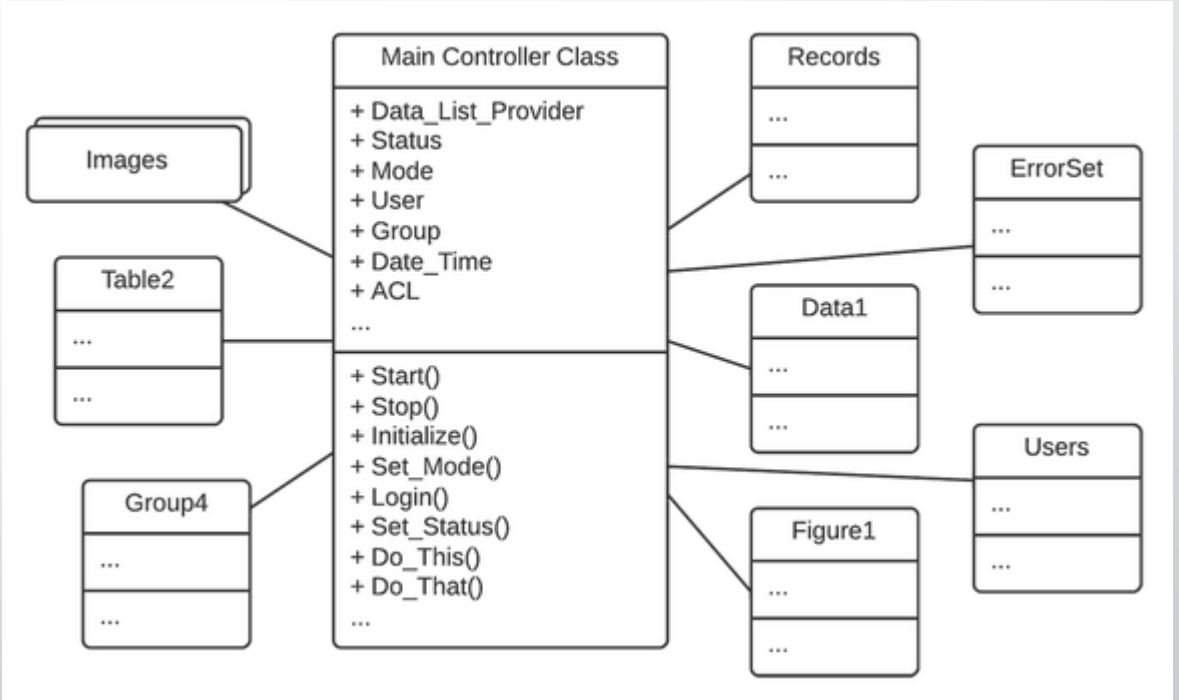
- Der Blob verschlingt die ganze Objekt-orientierte Architektur!

Beschreibung

- Anekdote: „Dies ist die Klasse, welche *wirklich* das Herz unserer Architektur ist!“

- Durch Prozedurales Design werden der Prozess und die Daten getrennt
- Eine Klasse ist für die Mehrheit aller Funktionalitäten verantwortlich!

- Eig. nur von Mikrocontroller bekannt, wo Performance wichtiger als Wartung ist



Symptome

- Große Anzahl an Attributen und/oder Methoden
- Kein durchgehender Zusammenhang zwischen den Attributen
- Nur eine komplexe Controller Klasse, ansonsten simple Daten-Objekt-Klassen
- Keine Kommunikation zwischen Objekten, nur Abhängigkeiten vom Gottobjekt/Blob
- Abwesenheit von Objekt-Orientierung

Auswirkungen

- Durch die Komplexität ist die Klasse schlecht zu warten und zu testen
- Der Blob ist anfällig für unnötigen/dead Code und macht es schwieriger diesen von nützlichem Funktionen zu trennen
- Durch die Zentralisierung der gesamten Funktionalität ist die Anwendung schlecht zu modifizieren
- Durch die Größe kostet die Klasse viel Ressourcen zum Laden

Ursachen

- Fehlende Erfahrung/Verständnis für Objektorientiertes Softwaredesign und abstrakte Modellierung
- Fehler beim Planen und Umsetzen der Architektur
  - Keine oder falsche Architektur definiert
  - Zu wenig/schlechte Reviews und kein Einschreiten wenn spezifizierte Architektur nicht eingehalten wird
  - Antipattern fließt fälschlicherweise als Anforderung in die Architektur ein
- Fehlendes Refactoring
- Migration von Legacy Codebases

Lösung

- Präventiv:
  - Know-How erweitern z.B. durch Schulungen oder Eigeninitiative
  - Die Art der Software Anforderungs-Spezifikation soll keinen negativen Einfluss auf die Software Architektur haben
  - Eine geeignete Software Architektur im Vorfeld planen, ermitteln und konsistent einhalten
- Refactoring of Responsibilities
  - Semantisch zusammengehörige Attribute und Operationen in eigene Klassen auslagern
  - Manche Operationen und Attribute können direkt in die zugehörigen einfachen Daten-Objekt-Klassen ausgelagert werden
  - Single responsibility / Seperation of Concerns wiederherstellen

Zusammenfassung / Template

- AntiPattern Name: The Blob
- Auch bekannt als: God Object, God Class, Winnebago
- Most Frequent Scale: Anwendung
- Refactored Solution Name: Refactoring of Responsibilities
- Refactored Solution Type: Software
- Root Causes: Faulheit, Hektik, schlechte Architekturplanung
- Unbalanced Forces: Verwaltung der Funktionalität, Performance, Komplexität
- Anecdotal Evidence: "This is the class that is really the *heart* of our architecture."

Quellen

- http://sourcemaking.com/antipatterns/the-blob
- http://www.antipatterns.com/briefing/sld024.htm
- http://en.wikipedia.org/wiki/God_object