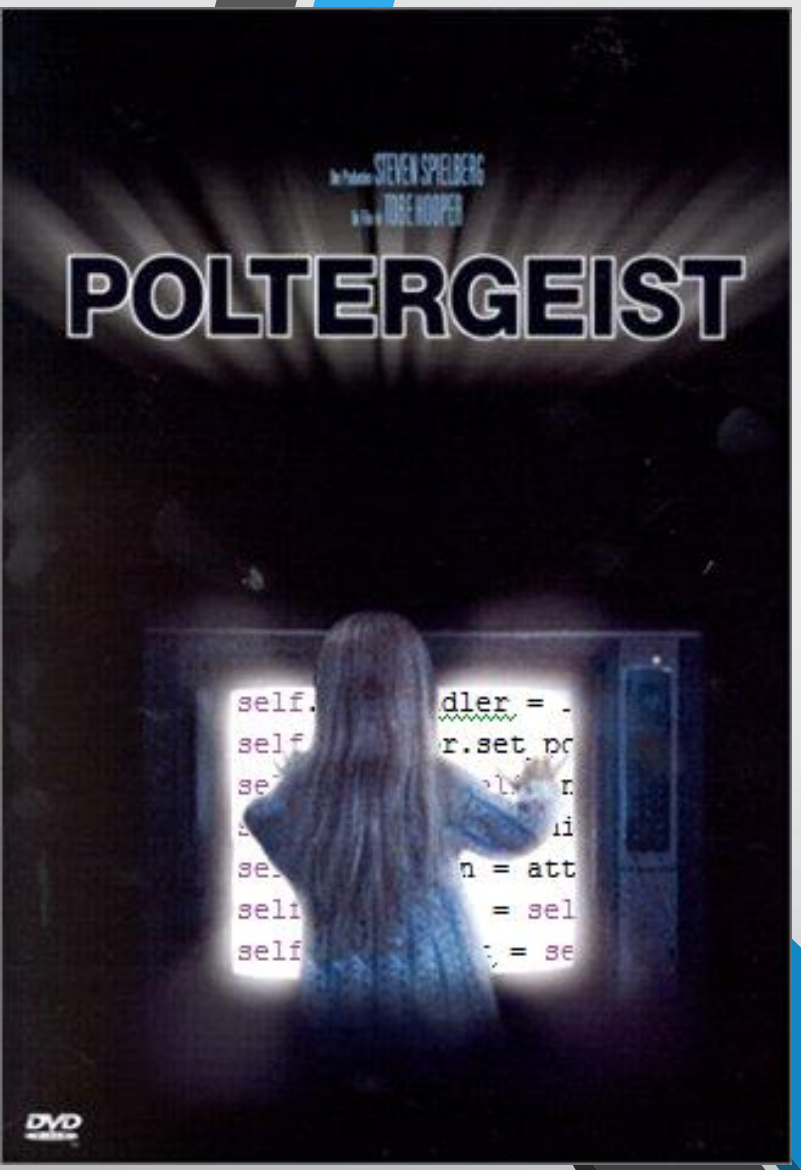


Poltergeist Anti-Pattern

Auch als Gypsy, Proliferation of Classes und Big Dolt Controller Class bekannt

Von Marco Wagner und Andreas Waigand

Am 16.10.2014

Agenda

- Beschreibung
- Auftreten
- Konsequenzen
- Lösung
- Beispiele
- Abgrenzung
- Zusammenfassung

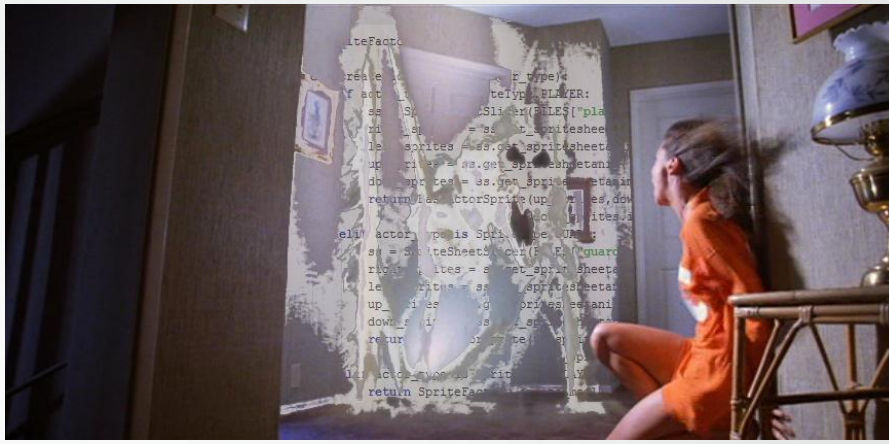
Beschreibung

- Poltergeist: eine Klasse mit stark eingeschränkten Rollen
- Startet meist nur Prozesse für andere Objekte
- Sehr kurzer Lebenszyklus
- Keine eigenen Zustände

Klassifizierung

- Kann als Architektur Anti Pattern definiert werden
- Entwicklung Anti Pattern passt eher, da der Poltergeist beim erstellen der Anwendung auftritt

Auftreten



- Häufigstes Auftreten: Bei der Erstellung von Anwendungen
- Wird meist als Controller eingeführt um Methoden anderer Klassen aufzurufen
- Auslöser ist meist ein falsches Verständnis von Objektorientierung oder Faulheit
- Oft wird auch fälschlicherweise der objektorientierte Ansatz gewählt obwohl dieser die Anforderungen nicht erfüllt

Symptome und Konsequenzen

- Verschwenden Ressourcen jedes Mal wenn sie „Auftauchen“
- Unnötig erhöhte Komplexität
- Verwaltung der Funktionalität wird schwieriger
- Ineffizient durch die Nutzung redundanter Navigationspfade
- Es entstehen Klassen ohne Zustände
- Man hört häufig den Satz:

Ich bin nicht ganz sicher was diese Klasse tut, aber sie ist aufjedenfall wichtig!“

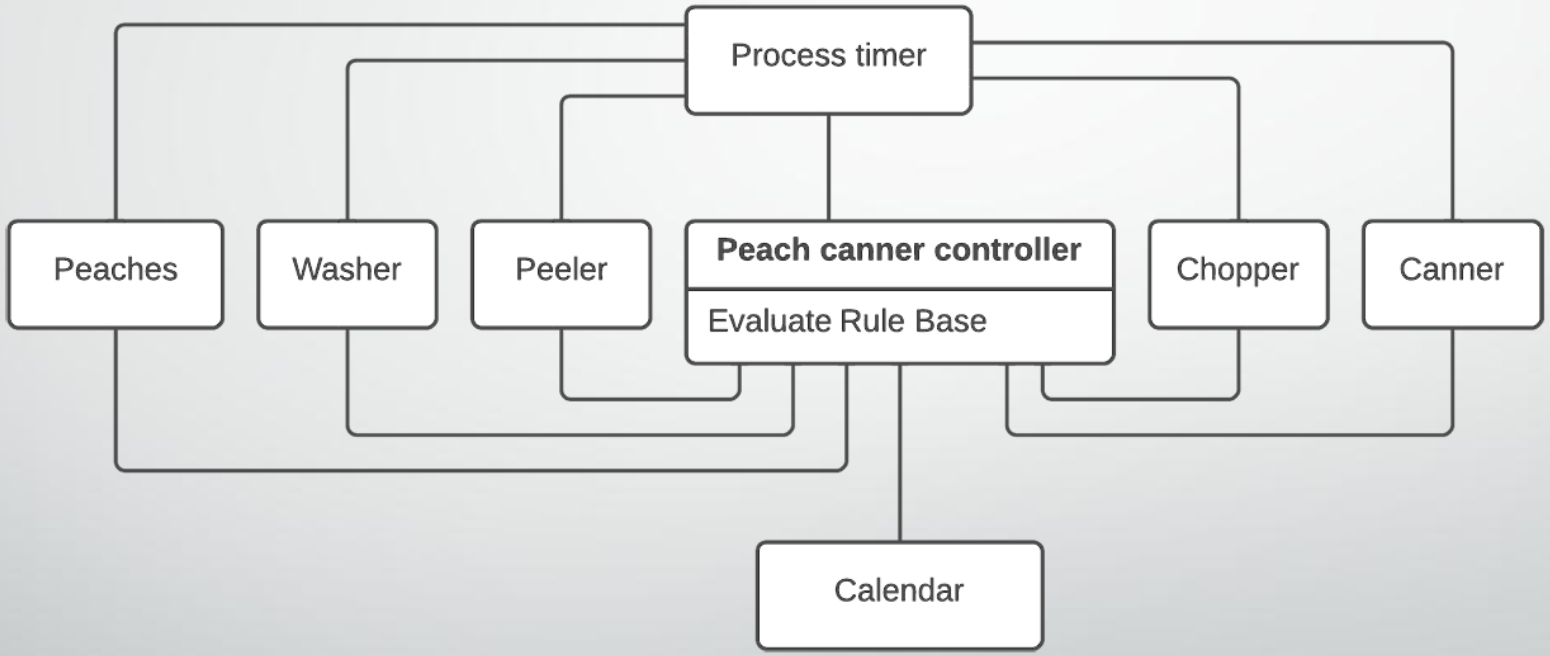
Lösung: “Ghostbusting“!

- Entfernung der Poltergeister
- Verlegung der Zuständigkeiten in langlebigeren Objekten

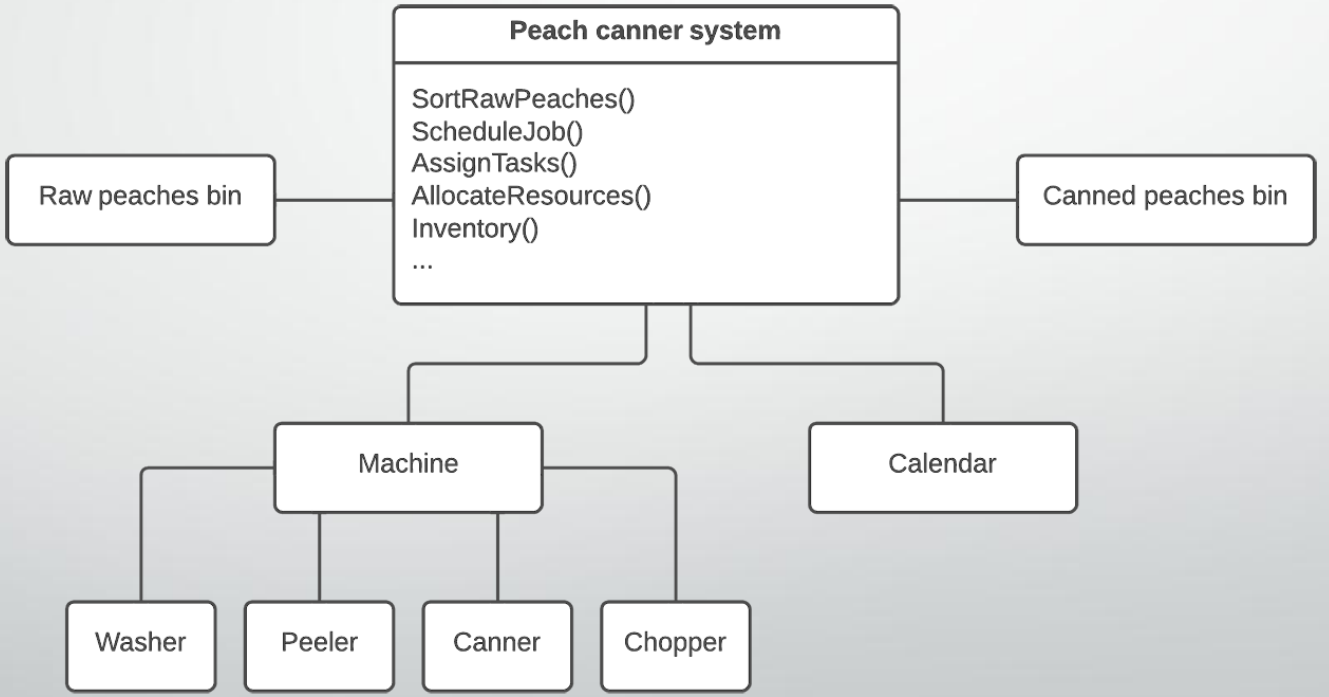
Vermeidung

- Am einfachsten lässt sich Poltergeist vermeiden indem frühzeitig erfahrene Architekten für den objektorientierten Ansatz eingebunden

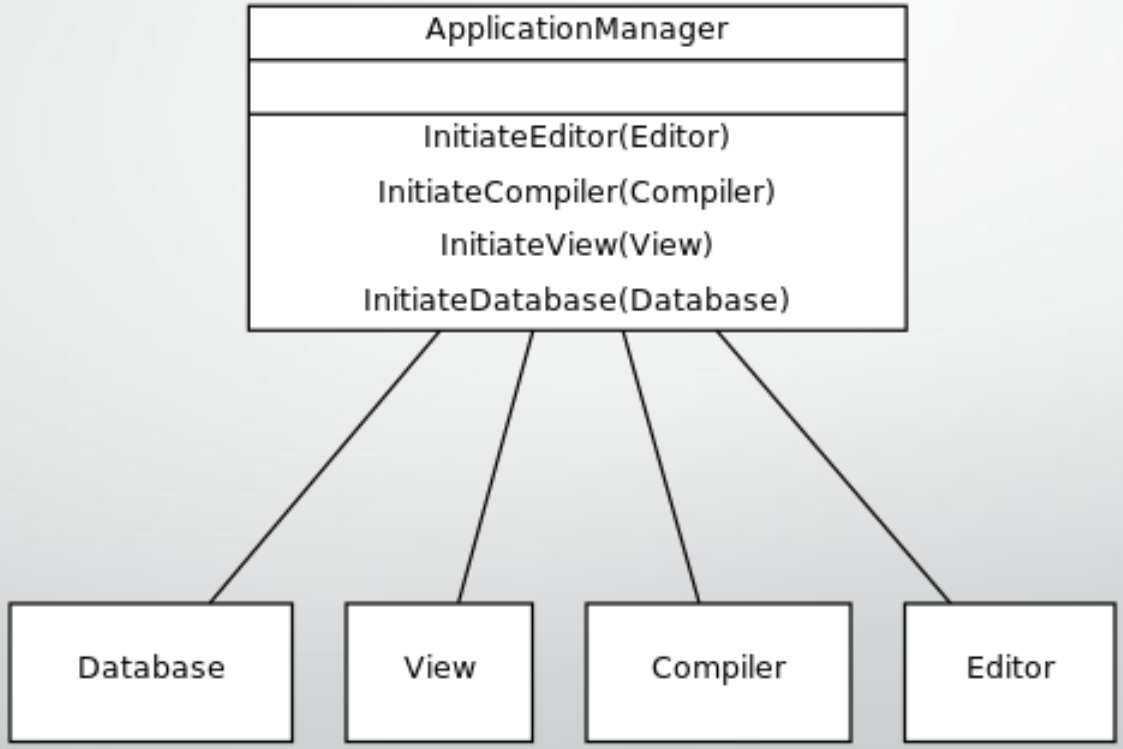
Beispiele


Process timer
Peaches
Washer
Peeler
Peach canner controller
Evaluate Rule Base
Chopper
Canner
Calendar

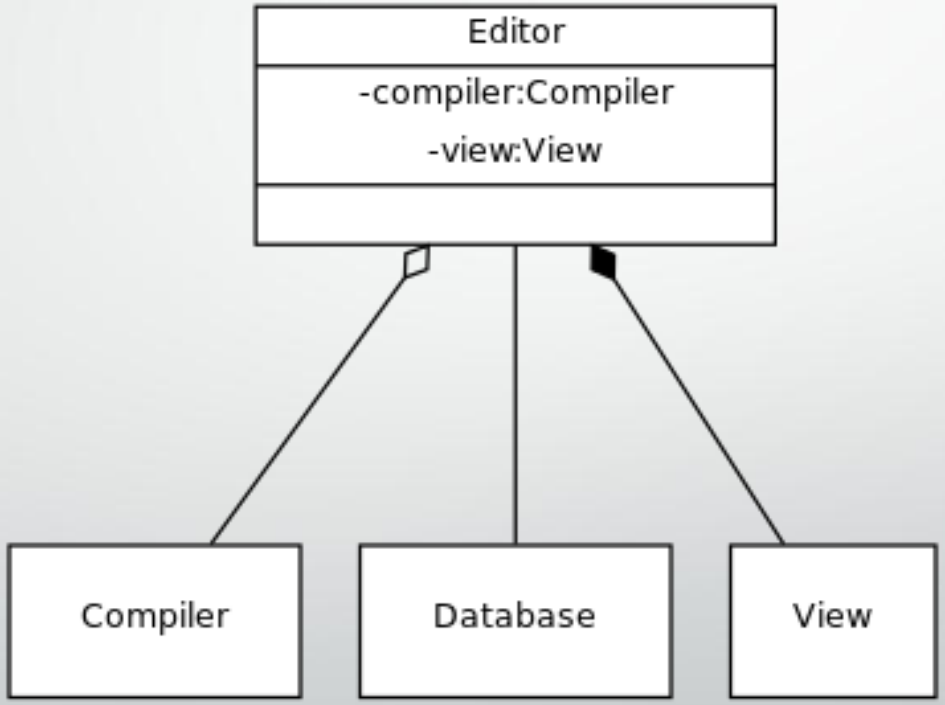
Beispiele


Peach canner system
SortRawPeaches()
ScheduleJob()
AssignTasks()
AllocateResources()
Inventory()
...
Raw peaches bin
Canned peaches bin
Machine
Calendar
Washer
Peeler
Canner
Chopper

Beispiele


ApplicationManager
InitiateEditor(Editor)
InitiateCompiler(Compiler)
InitiateView(View)
InitiateDatabase(Database)
Database
View
Compiler
Editor

Beispiele


Editor
-compiler:Compiler
-view:View
Compiler
Database
View

Beispiele

```
ublic class BaggageStore {

   Storage storage = new Storage();

   public void storeBaggage(Baggage bag){
       ReclaimController rc = new ReclaimController();
       rc.prepareStorage(storage);
       storage.store(bag);
   }
```

Beispiele

```
olic class BaggageStoreF {

  Storage storage = new Storage();

  public void storeBaggage(Baggage bag){
      storage.prepare();
      storage.store(bag);
  }
```

Abgrenzung

- Ein Poltergeist hat Ähnlichkeiten zu einem Controller (z.B. aus MVC):
  - Controller Lebenszyklus ist länger
  - Controller hat Zustände
- Ein Poltergeist hat Ähnlichkeiten zu einer Factory (besonders Simple Factory, da zustandslos):
  - Factory dient einzig zur Objekterzeugung
  - Factory hat längeren Lebenszyklus

Zusammenfassung

- AntiPattern Name: Poltergeist
- Auch bekannt als: Gypsy, Proliferation of Classes, und Big Dolt Controller Class
- Most Frequent Scale: Anwendung
- Refactored Solution Name: Ghostbusting
- Refactored Solution Type: Process
- Root Causes: Faulheit, Unwissen
- Unbalanced Force: Managament of Functionality, Management of Complexity
- Satz-den-man-häufig-hört: Ich bin nicht ganz sicher was diese Klasse tut, aber sie ist auf jedenfall wichtig!

Quellen

- Mowbray, Thomas J. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. Auflage: 1. Wiley, 2008.
- Freeman, Eric, Elisabeth Robson, Bert Bates, and Kathy Sierra. *Head First Design Patterns*. Auflage: 2. Sebastopol, CA: O'Reilly & Associates, 2006.
- http://sourcemaking.com/antipatterns/poltergeists

Fragen?