Bericht Abschlussprojekt

1 Einleitung

In meinem Abschlussprojekt habe ich untersucht, inwiefern ein Zusammenhang zwischen der Dichte der Bad Smells in Softwareprojekten mit der Anzahl der Autoren zusammenhängt. Dafür habe ich die folgenden Hypothesen überprüft:

1. Projekte mit einer größeren Anzahl an Autoren haben eine höhere Bad-Smell-Dichte. (between-subject)
2. In Projekten korrelliert die Bad-Smell-Dichte mit der Anzahl der Autoren. (within-subject)

Im Folgenden beschreibe ich zunächst meine Vorgehensweise und gehe dann auf die Ergebnisse meiner Auswertung ein.

2 Vorgehensweise

Grundsätzlich habe ich BOA [2] zur Datensammlung verwendet. Daher war der erste Schritt, Bad Smells zu identifizieren, die sich über die in BOA erhebbaren Metriken erkennen lassen. Hierbei handelt es sich um Bad Smells, die quantifizierbar sind: Viele Bad Smells, wie z.B. Unangebrachte Intimität, lassen sich nicht direkt aus einer Analyse des Syntaxbaums ableiten und können daher nicht mithilfe von BOA erfasst werden.

Es gibt nach Fowler keine festgelegten Kennzahlen, wann zum Beispiel eine Methode zu viele Parameter oder zu viele Codezeilen enthält [1], daher habe ich mir hier eigene Grenzwerte überlegt, die ich passend finde. Diese Grenzwerte können für spätere Experimente auch nochmal angepasst werden.

Folgende Bad Smells habe ich in diesem Projekt betrachtet:

- **Large Class:** Große Klassen deuten darauf hin, dass die Klasse in mehrere Klassen mit eigenen Zuständigkeiten zerlegt werden könnte. [1] Als große Klasse definiere ich eine Klasse mit mehr als 15 Methoden oder mehr als 10 Attributen.
- **Long Method:** Lange Methoden sind unübersichtlich und schwer zu verstehen, was es Entwicklern später schwer macht, deren Funktion nachzuvollziehen. [1] Als lange Methode definiere ich eine Methode, die aus mehr als 15 Statements besteht oder deren Zyklomatische Komplexität 10 übersteigt. Eine hohe Zyklomatische Komplexität deutet auf einen zu komplexen Kontrollfluss hin, der ebenfalls schwer nachzuvollziehen ist. Auch kommt es oft vor, dass solche Methoden duplizierten Code enthalten. [1]
- **Long Parameter List:** Methoden mit vielen Parametern sind oft verwirrend und sollten daher vermieden werden. [1] Ich betrachte in diesem Fall Methoden, die mehr als 4 Parameter haben.

2.1 Implementierung des BOA-Skripts

Bei der Implementierung des BOA-Skriptes habe ich mit mehreren Abstraktionsebenen gearbeitet, um so einen besser lesbaren und auch für mich selbst besser verständlichen Code zu erhalten. Zuerst habe ich einige Hilfsfunktionen definiert, die mich nach und nach ans Ziel gebracht haben.

2.1.1 Erste Abstraktionsebene

Auf der untersten Abstraktionsebene habe ich Funktionen definiert, die die für mich relevanten Kennzahlen (Anzahl Klassenfelder und -methoden, Zyklomatische Komplexität einer Methode, Anzahl von Statements einer Methode, Anzahl der Parameter einer Methode) bestimmen. Dafür habe ich meist Visitoren in Kombination mit if-Bedingungen verwendet, um so je eine Zählvariable zu erhöhen. Zum Zählen der Parameter konnte ich einfach die `len()`-Funktion verwenden. Außerdem habe ich eine Funktion zum zählen aller Autoren implementiert, die bis zu einem bestimmten Datum an einem Projekt mitgearbeitet haben; diese Funktion zähle ich auch zur ersten Abstraktionsebene.

2.1.2 Zweite Abstraktionsebene

Die zweite Abstraktionsebene besteht aus Funktionen, deren Rückgabetyp je ein boolscher Wert ist und die die Kriterien der jeweiligen Bad Smells überprüfen: Je eine Methode zum Erkennen großer Klassen, langer Methoden und langer Parameterlisten. Dafür werden die Rückgabewerte der Funktionen der ersten Ebene jeweils mit den definierten Grenzwerten abgeglichen.

2.1.3 Dritte Abstraktionsebene

Auf der dritten Abstraktionsebene gibt es eine einzige Funktion, die ein `CodeRepository` und eine Jahreszahl als Parameter erhält, und dann alle nötigen Kennzahlen für diese berechnet und in den Output schreibt. Hier wird in einem Visitor über einen Snapshot für das übergebene Jahr iteriert und je Zählvariablen für Bad Smells in Klassen, Gesamtzahl der Klassen, Bad Smells in Methoden und Gesamtzahl der Methoden hochgezählt. Schlussendlich werden all diese Zahlen zusammen mit der Anzahl der Autoren in einem String in den Output geschrieben, mit Projektname und Jahreszahl als Index.

Die Hauptroutine ist ein Visitor, der für jedes `CodeRepository` die Kennzahlen für die Jahre 2000 bis 2022 mithilfe dieser Funktion berechnet.

2.2 Verwendete Datensätze

Während der Entwicklung des Skriptes habe ich den Datensatz *2019 October/GitHub (small)* verwendet, daher wollte ich, um eine möglichst große Variation verschiedener Softwareprojekte zu betrachten, den Datensatz *2019 October/GitHub* für die endgültige Datenerhebung verwenden. Zunächst sah es auch so aus, als würde die Anfrage korrekt bearbeitet werden, allerdings war der Job nach ca. 20 Stunden Wartezeit immer noch nicht abgeschlossen. Unter *Job Status* wurde *map() completion: 1.0; reduce() completion: 0.0* angezeigt, somit denke ich, dass es hier intern zu einem Fehler gekommen sein muss.

Bei den Datensätzen *2022 Jan/Java* und *2022 Feb/Python* endete die Berechnung nach etwa zwei Stunden mit einem Error, der nicht weiter beschrieben wird. Das gleiche Phänomen ist bereits in der Übung aufgetreten.

Einzig der Datensatz *2019 October/GitHub (medium)* lief erfolgreich durch und hält eine ausreichende Sample Size bereit, um ein signifikantes Ergebnis zu erzielen. Somit habe ich die Berechnungen letztendlich mit diesem Datensatz durchgeführt.

2.3 Auswertung

Für die Auswertung habe ich die Bad-Smell-Dichte auf Klassenebene und die Bad-Smell-Dichte auf Methodenebene unabhängig voneinander betrachtet und alle Tests mit beiden Werten ausgeführt, da sie sich in meinen Augen nicht in einen Topf werfen lassen: Da ein Projekt meist weniger Klassen als Methoden hat, wären die Bad Smells auf Klassenebene sonst unterrepräsentiert.

Für die Auswertung habe ich Python mit den Bibliotheken *Pandas, Numpy, Scipy und Statsmodels* verwendet. Folgende statistische Tests habe ich auf die Daten angewandt:

- Between-Subject Analyse:
  - **Pearson-Korrelation**: Zur Untersuchung des linearen Zusammenhangs zwischen der durchschnittlichen Anzahl der Autoren und der Bad Smell Density (für Klassen und Methoden) über verschiedene Projekte hinweg.
  - **Lineare Regression**: Um den Einfluss der Autorenanzahl auf die Density zu modellieren und den Anteil der erklärten Varianz (R^2) zu quantifizieren.

  Warum: Diese Methoden ermöglichen es, aggregierte Zusammenhänge auf Projektebene zu analysieren und einen direkten Effekt der Autorenanzahl zu bestimmen.
- Within-Subject Analyse:
  - **Mixed-Effects Modell**: Zur Analyse des Einflusses der Autorenanzahl auf die Bad Smell Density innerhalb einzelner Projekte über die Zeit, wobei projektspezifische Unterschiede als Zufallseffekte berücksichtigt werden.

  Warum: Dieses Modell ist geeignet, um Veränderungen innerhalb der Projekte zu untersuchen und individuelle Unterschiede zwischen den Projekten zu kontrollieren.

3 Ergebnisse und Interpretation

Folgende Ergebnisse habe ich bei der Statistischen Auswertung der Daten aus dem Datensatz *2019 October/GitHub (medium)* erzielt:

3.1 Between-Subject Analyse

- Bad Smells auf Klassenebene:
  - Pearson-Korrelation: $r = 0.057$ ($p < 0.001$) und Regressionskoeffizient = 0.0011 zeigen einen statistisch signifikanten, aber äußerst schwachen positiven Zusammenhang zwischen der durchschnittlichen Autorenanzahl und der Dichte von Klassenbezogenen Bad Smells.
  - Der R^2-Wert von 0.003 bedeutet, dass lediglich ca. $0,3\%$ der Varianz in der Dichte durch die Autorenanzahl erklärt werden.
- Bad Smells auf Methodenebene:
  - Pearson-Korrelation: $r = 0.028$ ($p < 0.001$) sowie ein Regressionskoeffizient von 0.0005 deuten ebenfalls auf einen statistisch signifikanten, jedoch praktisch vernachlässigbaren positiven Zusammenhang hin.
  - Hier erklärt die Autorenanzahl nur ca. $0,1\%$ der Varianz ($R^2 = 0.001$).

3.2 Within-Subject Analyse

- Bad Smells auf Klassenebene:
  - Das Mixed-Effects Modell ergibt einen negativen Effekt (Koeffizient ca. -0.000, $z = -5.613$, $p < 0.001$).
  - Das heißt, wenn innerhalb eines Projekts die Autorenanzahl ansteigt, sinkt – wenn auch minimal – die Klassen-Density.
- Bad Smells auf Methodenebene:
  - Auch hier zeigt das Modell einen negativen Effekt (Koeffizient ca. -0.000, $z = -29.843$, $p < 0.001$), was auf einen sehr geringen Rückgang der Methoden-Density bei steigender Autorenanzahl hindeutet.

4 Zusammenfassung

Zusammenfassend lässt sich sagen, dass kein bedeutender Zusammenhang (Korrelation) zwischen der Anzahl der Autoren in einem Softwareprojekt und der Dichte der Bad Smells nachgewiesen werden konnte: Weder zwischen verschiedenen Softwareprojekten, noch innerhalb eines Projekts. Man muss jedoch beachten, dass hier nur ein kleiner Teil der tatsächlich vorhandenen Bad Smells überhaupt betrachtet wurde; Bei einer detaillierteren und umfassenderen Analyse könnte das Ergebnis gegebenenfalls anders aussehen.

Literatur

[1] Martin Fowler. *Refactoring: improving the design of existing code.* Addison-Wesley Professional, 2018.

[2] Iowa State University. Boa language and infrastructure. `https://boa.cs.iastate.edu/` [Accessed: March 13th 2025].