

Komplexität von Refactoring im Kontext

Verteidigung und Diskussion

Übersicht

- Einführung: Begriffe und Ansätze
- Stand der Literatur
- Fragestellungen
- Vorgehen
- Ergebnisse
- Ausblick

Einführung: Refactoring

Eine Verbesserung der internen Struktur des Quellcodes, ohne das externe Verhalten zu ändern.

- Was ist extern, was intern?
- Gibt es Zusammenhänge zu Verhaltensänderungen?
- Wer veranlasst das Refactoring?

Einführung: Ansatz der Untersuchung

- Möglichkeiten nach Murphy-Hill 2008:
 1. Analyse der Commit-Nachrichten
 2. Analyse der Code-Änderungen
 3. Beobachtung von Programmierepisoden
 4. Mitschreiben der Verwendung von Werkzeugen
- Ansatz 2 erlaubt nachträgliche Analyse im Detail
 - Großer Datensatz erfordert starke Einschränkung der Suche
 - Weniger Abhängig von der Wahl der Projekte
 - Ich kann bisher nicht betrachtete Aspekte hervorheben

Stand der Literatur

- Diverse systematische Literaturübersichten (SLR), sogar tertiäre
 - Häufig Anwendung neuer Werkzeuge oder Metrik auf festen Softwarestand
- Analyse der Historie
 - Commit-Nachrichten: AlOmar et al. 2019-2021
 - Änderungen des Quellcodes
 - Werkzeuge: Ref-Finder (Kim et al. 2010) und RefactoringMiner (Silva et al. 2016)
 - Darauf aufbauend: Pantiuchina et al. 2020

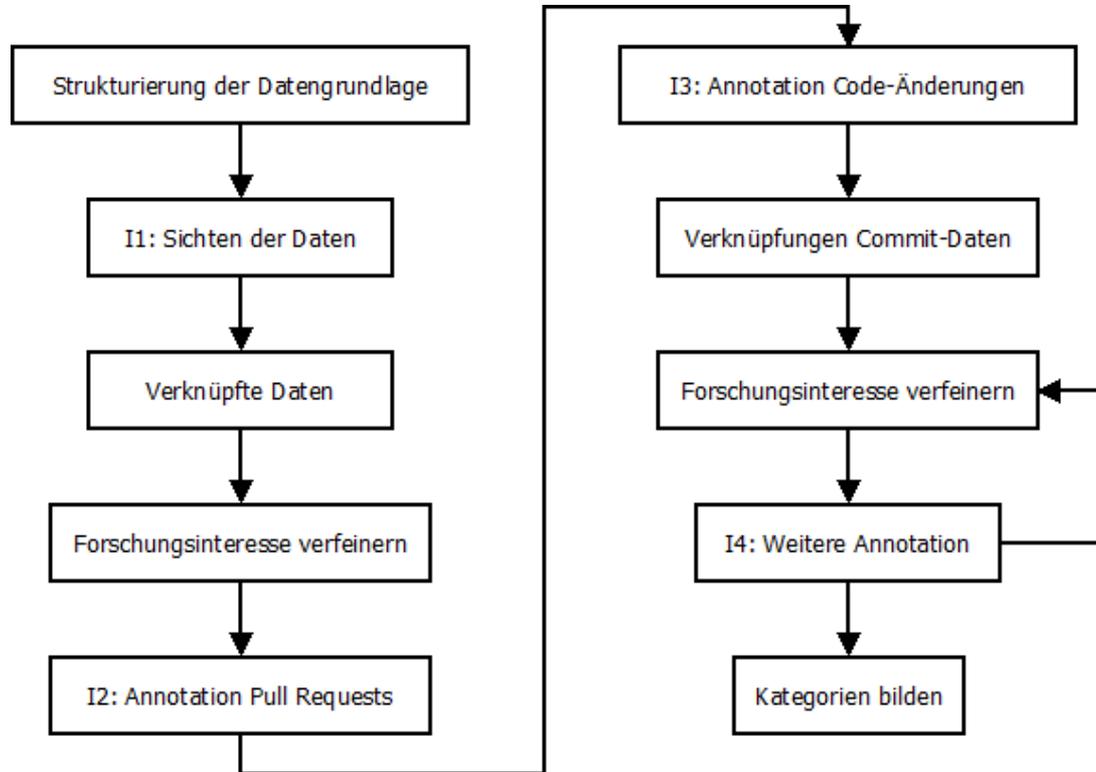
Fragestellungen

- Gibt es verschiedene Arten von Pull Requests, welche bisher nicht unterschieden wurden?
 - Welche Kriterien sind dafür relevant?
- Wie entwickelt sich die Diskussion in diesen Fällen?
- Welche Hindernisse offenbaren sich bei der Untersuchung?

Die richtigen Fragen

- Herausforderung für mich: Überblick gewinnen
 - über bereits bearbeitete Fragestellungen
 - über mögliche Lücken
 - über angebrachte Systematik für das Vorgehen
- Es ist schwierig, die richtigen Fragen zu stellen, um die richtigen Antworten zu bekommen

Vorgehen: Prozess



Vorgehen: Kontext von Pull Requests

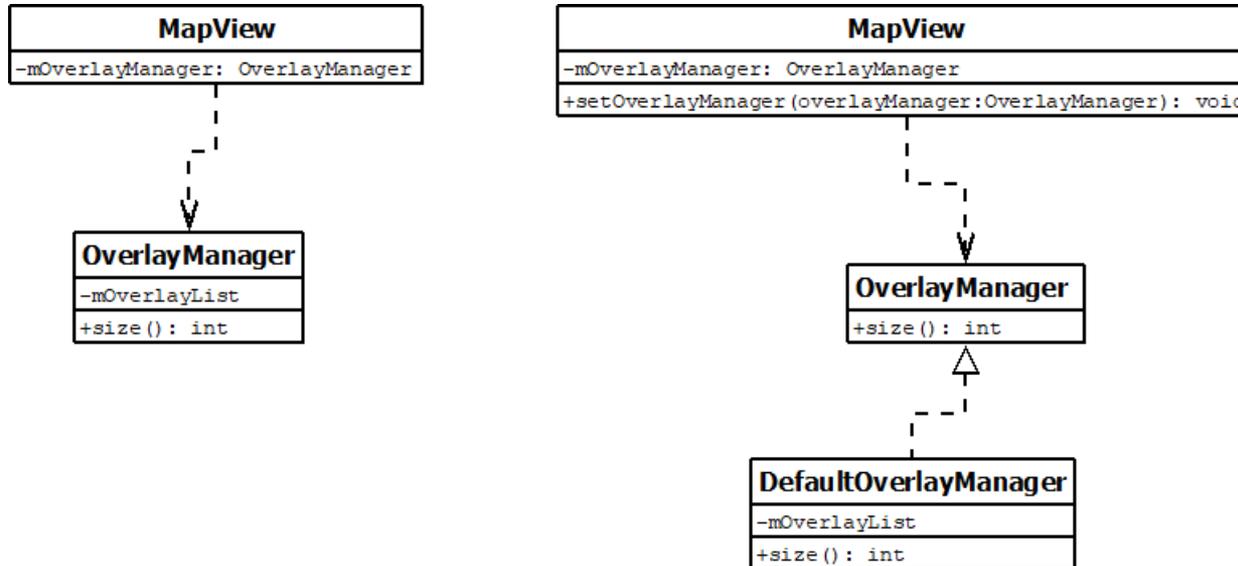
- Verknüpfte Issues (Titel, Commits, Verlinkung)
- Externe Dokumentation (z.B. Confluence)
- Inhalt und Interaktion der Diskussion (Beteiligte, Gegenstand)
- Baum der Änderungen in git
(vorherige und nachlaufende Änderungen)
- Metadaten der Commits (Datum, Ersteller, Nachricht)

Ergebnisse: Reines Refactoring

- Pull Request ausschließlich mit Refactoring
 - Keine Änderung externen Verhaltens
 - Strukturverbesserung des Quellcodes

- Kann im Zusammenhang zu anderen externen Änderungen stehen

Reines Refactoring: Beispiel



Reines Refactoring?

- Vorheriges Beispiel: PR 179 aus Projekt osmdroid
- Streng genommen ist MapView aber Teil der öffentlichen Schnittstelle
-> externes Verhalten
- 3 weitere Beispiele mit kleineren Abweichungen

```

4  osmdroid-android/src/main/java/org/osmdroid/util/TileSystemWebMercator
   @@ -6,8 +6,8 @@
6   */
7   public class TileSystemWebMercator extends TileSystem{
8
9   -   private static final double MinLatitude = -85.05112877980659;
10  +   private static final double MinLatitude = -85.05112877980658;
11  -   private static final double MaxLatitude = 85.05112877980659;
12  +   private static final double MaxLatitude = 85.05112877980658;
13  +   private static final double MinLongitude = -180;
14  +   private static final double MaxLongitude = 180;

```

Reines Refactoring: Eigenschaften

- Wenig Diskussion zu den vorliegenden Fällen
 - 2x Nachfrage zur konkreten Umsetzung
 - 2x zusätzlicher, nicht themenverwandter Vorschlag
 - Struktur selbst nicht infrage gestellt
- Aufteilung der Pull Requests ist zwischen Projekten unterschiedlich
 - Bei osmdroid werden die PR kleiner aufgeteilt
 - DSpace hat meist einen PR pro Jira-Ticket, damit alle Änderungen zusammen

Ausblick

- Vertiefung Reine Refactorings
 - Breitere Datenbasis mit mehreren Projekten
 - Bessere Abgrenzung in Graubereichen
- Verbesserte Zuordnung von Kontext zu Pull Request
 - Automatisierte Verknüpfung der Issues
 - Berücksichtigung Erstellungsdatum von Commits
- Verwendung des Begriffs unter Entwicklern

Komplexität von Refactoring im Kontext

DISKUSSION

Backup: Fachwörter Gitlab

Pull Request

DSpace refactored service api #1021

Merged mwoodiupui merged 1 commit into DSpace:DS-2781-service-api from KevinVdV:dSPACE-service-api on Aug 19, 2015

Conversation 2 Commits 1 Checks 0 Files changed 1,145

KevinVdV commented on Aug 19, 2015

Pull request for the service based api, see <https://wiki.duraspace.org/display/DSpace/DSpace+Service+based+api> & <https://jira.duraspace.org/browse/DS-2701> for more information

DSpace refactored service api 54222f3

mwoodiupui added a commit that referenced this pull request on Aug 19, 2015

Merge pull request #1021 from KevinVdV:dSPACE-service-api 758d8ab

mwoodiupui merged commit 758d8ab into DSpace:DS-2781-service-api on Aug 19, 2015

peterdietz reviewed on Aug 20, 2015

```

dSPACE-API/src/main/java/org/dspace/app/util/AuthorizeUtil.java
418 -     authorizeManager.authorizeAction(context, collection
419 -     .getCommunities()[0], Constants.ADMIN);
422 +     authorizeService.authorizeAction(context, collection
423 +     .getCommunities().get(0), Constants.ADMIN);
    
```

peterdietz on Aug 20, 2015

Is this the proper Service way, or needs another pass?
collection.getCommunities().get(0)

Commit

DSpace refactored service api

main (#1021)
dSPACE-7.4 ... dSPACE-6.0-rc1

KevinVdV committed on Aug 19, 2015

14 build.properties

```

@@ -56,6 +56,7 @@ default.language = en_US
56 56 # Uncomment the appropriate block below for your database.
57 57 # postgres
58 58 db.driver=org.postgresql.Driver
59 59 + db.dialect=org.dspace.storage.rdbms.hibernate.postgres.DSPACEPostgreSQLDialect
60 60 db.url=jdbc:postgresql://localhost:5432/dspace
61 61 db.username=dSPACE
61 62 db.password=dSPACE

@@ -77,18 +78,9 @@ db.schema =
77 78 # Maximum number of DB connections in pool
78 79 db.maxconnections = 30
79 80
80 80 - # Maximum time to wait before giving up if all connections in pool are busy (milliseconds)
81 81 - db.maxwait = 5000
82 82 + # Determine the number of statements that can be cached (set to 0 to disable caching)
82 83 + db.statementpool.cache = 50
83 83
84 84 - # Maximum number of idle connections in pool (-1 = unlimited)
84 85 - db.maxidle = -1
85 85 -
86 86 - # Determine if prepared statement should be cached. (default is true)
87 87 - db.statementpool = true
88 88 -
89 89 - # Specify a name for the connection pool (useful if you have multiple applications sharing Tomcat
    
```

Backup: Datum eines Commits

Assume the following history exists and the current branch is "topic":

```

      A---B---C topic
      /
D---E---F---G master
    
```

From this point, the result of either of the following commands:

```

git rebase master
git rebase master topic
    
```

would be:

```

      A'---B'---C' topic
      /
D---E---F---G master
    
```

Diff
 Old version
 New version
Lines of context:
 Ignore s

Author: KevinVdV <kevin@mire.be> 2014-11-08 09:19:09
 Committer: KevinVdV <kevin@mire.be> 2015-08-19 11:54:32
 Parent: [fcb3717aad396924f8ddbca069f98acd3c6dd414](#) (Merge pull re
 Child: [758d8abdadee1979046d7ea6241956b4c2a23ec9](#) (Merge pull re
 Branches: master,

Quelle: Git Dokumentation git-rebase

Quellen

- Bild Backup PullRequest: GitHub, URL:
<https://github.com/DSpace/DSpace/pull/1021>
- Bild Backup Commit: GitHub, URL:
<https://github.com/DSpace/DSpace/pull/1021/commits/54222f3c1d3c63bd03af263703a950e9cfad7439>
- Bild Backup git-rebase: git-rebase Dokumentation. URL:
<https://git-scm.com/docs/git-rebase>