|  |
| --- |
| Elmer Lukas, Heidt Christina, Steiner Diego, Treichler Delia, Waltenspül Remo  30. Mai 2011 |

|  |
| --- |
| SE2 Projekt MRT |
| Codereview |
| Informales Codereview |

****

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 04.05.2011 | 1.0 | Erste Version des Dokuments | WR |
| 27.05.2011 | 1. | Korrekturen | TD |

## Inhaltsverzeichnis

[1 Dokumentinformationen 1](#_Toc294256519)

[1.1 Änderungsgeschichte 1](#_Toc294256520)

[1.2 Inhaltsverzeichnis 1](#_Toc294256521)

[2 Einführung 2](#_Toc294256522)

[2.1 Zweck 2](#_Toc294256523)

[2.2 Gültigkeitsbereich 2](#_Toc294256524)

[2.3 Definitionen und Abkürzungen 2](#_Toc294256525)

[2.4 Referenzen 2](#_Toc294256526)

[3 Codestatistiken 3](#_Toc294256527)

[3.1 Android 3](#_Toc294256528)

[3.2 Rails 3](#_Toc294256529)

[4 Codemetriken 4](#_Toc294256530)

[4.1 Android 4](#_Toc294256531)

[4.2 Rails 4](#_Toc294256532)

[5 Codereview 5](#_Toc294256533)

[5.1 Ablauf 5](#_Toc294256534)

[5.2 Fokus bei Review 5](#_Toc294256535)

[5.3 Eingesetzte Tools 5](#_Toc294256536)

[5.3.1 Findbugs 5](#_Toc294256537)

[5.3.2 Checkstyle 5](#_Toc294256538)

[5.3.3 State Of Flow Eclipse Metrics Plugin 5](#_Toc294256539)

[5.4 Android 6](#_Toc294256540)

[5.4.1 Review 1 (04.05.2011) 6](#_Toc294256541)

[5.5 Rails 7](#_Toc294256542)

[5.5.1 Review 1 (08.05.2011) 7](#_Toc294256543)

[6 Deckung mit der Planung 8](#_Toc294256544)

[6.1 Design 8](#_Toc294256545)

[6.2 Coderichtlinien 8](#_Toc294256546)

[6.2.1 Anpassungen 8](#_Toc294256547)

# Einführung

## Zweck

Der Zweck dieses Dokuments ist die Aufzeigung von Codestatistiken und -metriken, Dokumentation der Reviews und Festhaltung der Änderungen gegenüber der Planung bezüglich Design und Coderichtlinien für das Projekt MRT (Mobile Reporting Tool).

## Gültigkeitsbereich

Dieses Dokument ist während der gesamten Projektdauer gültig (21.02 bis 03.06.2011).

## Definitionen und Abkürzungen

Die Definitionen und Abkürzungen befinden sich in der ausgelagerten Datei doc/01\_Projektplan/glossar.docx.

## Referenzen

* doc/05\_design/metrics\_server/\*
* doc/05\_design/metrics/\*
* doc/05\_design/structure101/\*
* doc/01\_projektplan/glossar.doc

# Codestatistiken

## Android

## Rails

# Codemetriken

{REMO}

Für die Metrikanalyse des implementierten Codes wurden unter Rails das Gem Metric\_fu und unter Android die Anwendungen State of Flow Eclipse Metrics sowie State 101 verwendet.

Die nachfolgend beschriebenen Metrik Indikatoren, wiederspiegeln nur die interessantesten Werte bzw. Auffälligkeiten. Um eine ausführliche Metrik Analyse zu erhalten, wird auf die unter dem Kapitel 2.4 Referenzen aufgeführten Ordner verwiesen.

## Rails

### Metric\_fu

### Schlussfolgerung

## Android

### State of Flow Eclipse Metrics

Das Eclipse-Plugin State of Flow Eclipse Metrics wird verwendet um diverse Code Qualitätsindikatoren wie zum Beispiel McCabe’s Cyclomatic Complexity, Efferent Coupling, Feature Envy, Lines of Code in Method etc. aus dem Code zu berechnen. Diese werden zudem laufend während der Entwicklung neu ausgewertet und direkt in der IDE als Marker angezeigt.

#### Cyclomatic Complexity

Wie aus der nachfolgenden Abbildung 1 - Übersicht Cyclomatic Complexity ersichtlich, ist die zyklomatische Komplexität relativ gering. Sämtliche Klassen befinden sich innerhalb des gewünschten Bereichs von eins bis vier.

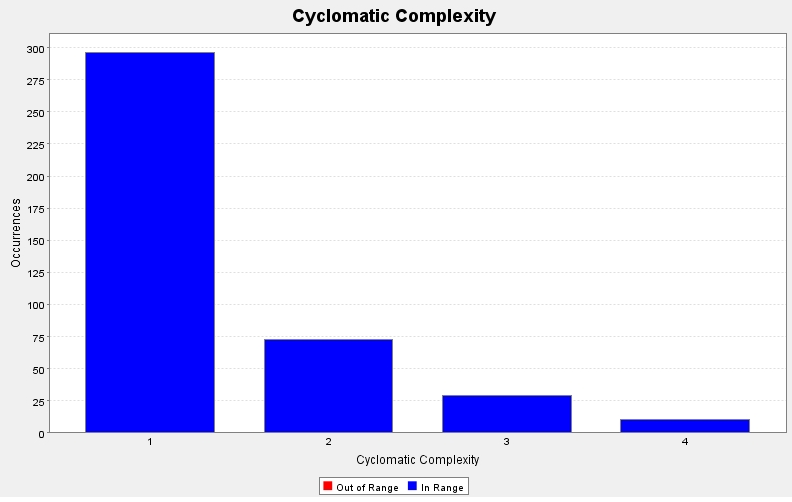


Abbildung - Übersicht Cyclomatic Complexity

#### Lines of Statements

Der Grenzwert für die Anzahl Anweisungen in einer Methode wurde auf 20 festgelegt. Bis auf wenige Ausnahmen wurde dieser Wert nicht überschritten (siehe Abbildung 2 - Übersicht Lines of Code). Bei den Methoden, welche nicht im definierten Bereich liegen, handelt es sich um Testmethoden.

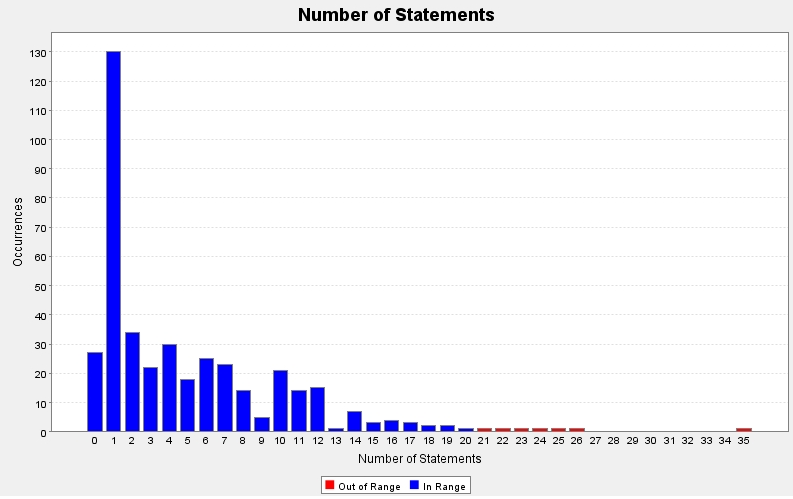


Abbildung - Übersicht Lines of Code

**Erklärung Long Method**

Bei der in Abbildung 3 ersichtlichen Methode testToJSON handelt es sich um die Methode mit den meisten Anweisungen. In der Gruppe wurde überlegt, diese Methode sowie weitere Testmethoden mittels des Refactoringprinzip „Extract Method“ weiter zu unterteilen. Dies wurde jedoch verworfen, aufgrund der nicht vorhandenen Wiederverwendbarkeit einer ausgelagerten Methode. Da jeder Test mit anderen Werten, Objekten initialisiert wird.

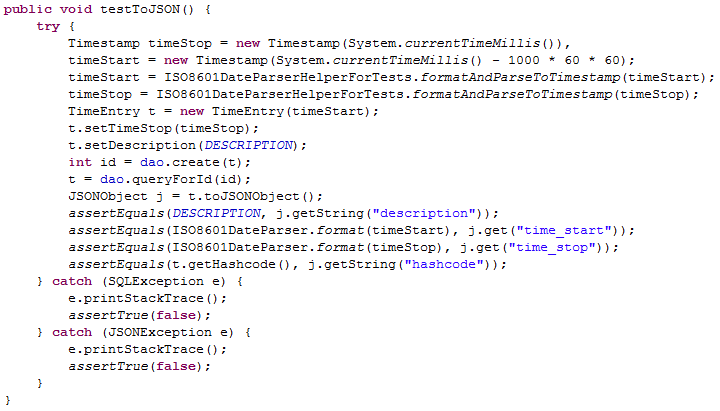


Abbildung - Methode mit den meisten Codezeilen

#### Lack of Cohesion in Methods (Total Correlation)

Bei der Codemetrik Lack of Cohesion wird geprüft, ob ein Muster beim Zugriff auf die Attribute existiert. Falls die Methoden nur einen Teil der Felder benutzen, verringert dies die Kohäsion.

Der definierte erwartete Wert sollte zwischen null und 50% liegen, wobei ein tieferer Wert für eine höhere Kohäsion steht.

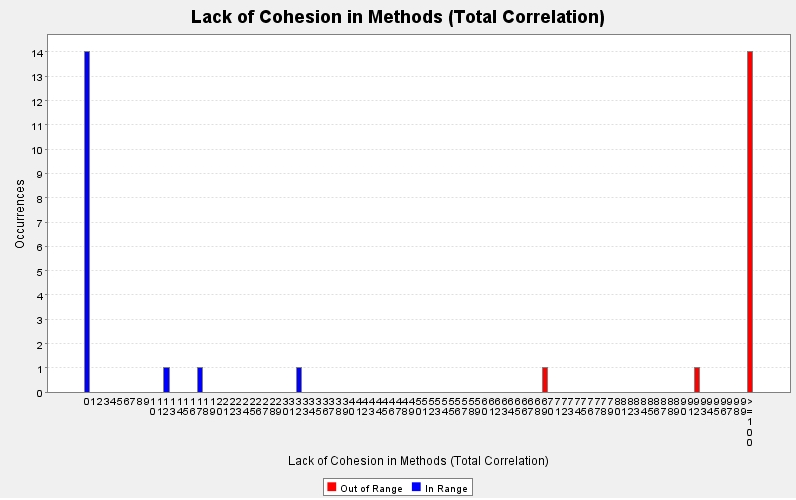


Abbildung - Übersicht Lack of Cohesion

**Erklärung tiefe Kohäsion**

Wiederum führen die Testklassen zu diesen hohen Werten bei der Analyse des Lack of Cohesion Indikator. Da die Testklasse ein breites Spektrum von verschiedenen Funktionalitäten testen muss, werden viele Attribute benötigt, welche jedoch von vielen Methoden gar nicht gebraucht werden.

Im Team wurde jedoch beschlossen, die Klasse nicht weiter zu extrahieren, da dies die Abstraktion erhöhen, sowie dem Grundsatz pro Klasse eine Testklasse widersprechen würde.

Die nachfolgende Abbildung 5 illustriert die Klasse TimeEntryActivityTest, welche einen eher geringen Kohäsionswert hat aufgrund der vielen Attribute.

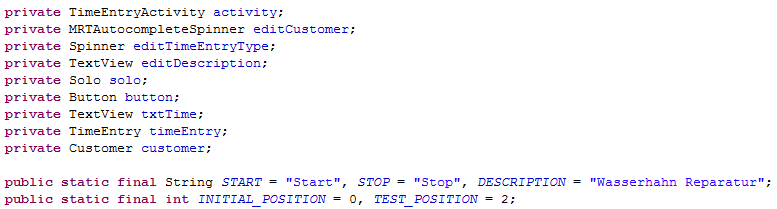


Abbildung - Viele Attribute in Klasse TimeEntryActivityTest

### State 101

### Schlussfolgerung

# Codereview

## Ablauf

Ein Review wird mindestens zu zweit durgeführt. Ein Teilnehmer des Reviews ist die Person, welche den ausgewählten Code geschrieben hat.

Der Review-Vorgang besteht aus dem Durchschauen des Codes und der Anwendung der unter Kapitel 3 „Eingesetzte Anwendungen, Plug-Ins“ erwähnten Software-Produkten.

Während des Reviews wird eine Liste geführt, in der alle gefundenen Ungereimtheiten und Fehler dokumentiert werden.

Nach dem Review üben die in der Liste zugewiesenen Personen die Fehlerbehebung oder das Refactoring für die bestimmten Codeabschnitte aus.

## Fokus bei Review

* Code auf Verständnis überprüfen
* Code Smells
* Fehler finden
* Namenswahl (aussagekräftige Namen für Methoden, Klassen, Packages)
* Korrekte Abbildung von Architekturentscheidungen

## Eingesetzte Tools

### Findbugs

Findbugs ist eine Open-Source Code-Analyse-Software, sie untersucht Java-Programme nach Fehlermustern. Es existiert ein Eclipse Plug-in für eine interaktive Nutzung.

Die Suche nach Fehlern geschieht ausschliesslich durch das Prüfen des Bytecodes, das Programm muss dazu nicht ausgeführt werden.

### Checkstyle

Checkstyle überprüft die Einhaltung von Coding Conventions bei der Erstellung von Java-Code. Convention-Verletzungen werden als Warnungen, nach Kategorie sortiert, angezeigt. Das Programm erlaubt einem, einen einheitlichen Coding Standard über alle Implementierungsarbeiten hinweg zu erreichen und so die Softwarequalität zu erhöhen.

### State Of Flow Eclipse Metrics Plugin

Das State Of Flow Eclipse Metrics Plugin berechnet verschiedenste Metriken bei der Überprüfung des Codes. Es gibt Warnungen für jede Metrik aus, bei der der Schwellwert überschritten ist.

* Das Exportieren eines HTML-Reports ist möglich, er bietet eine gute Übersicht über alle Warnungen pro Metrik des überprüften Codes.

## Reviews Android

### Review 1 (04.05.2011)

Review der Klasse LoginActivity

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Beschreibung | Schweregrad | Datum & Kürzel wenn behoben | Req. Ref. | Bemerkungen |
| 1 | Methode processLogin() ist zu umfangreich (smell Long Method) | Mittel | 04.05.2011 WR | - | Refactoring ExtractMethod |
| 2 | Ähnlicher Code wird mehrmals aufgerufen (smell Duplicated Code) | Mittel | 04.05.2011 WR | Ja | Refactoring ExtractMethod |
| 3 | Attributnamen zu wenig aussagekräftig | Leicht | 04.05.2011 WR | - | Refactoring RenameField |
| 4 | Name für Methode „checkPref“ zu wenig aussagekräftig | Leicht | 04.05.2011 WR | - | Refactoring RenameMethod |
| 5 | Zugriffsmodifikator für Methode switchToTimeEntryActivity() ist auf protected gesetzt | Mittel | 04.05.2011 WR | Ja | - |

#### Referenz ID 2

Die Objekte, welche die GUI Schaltflächen repräsentieren, werden bei jedem Zugriff neu geladen, anstatt diese direkt als Attribute der Klasse zu speichern.

#### Referenz ID 5

Die Methode switchToTimeEntryActivity() wird nur von der eigenen Klasse verwendet, dennoch steht der Zugriffsmodifikator auf protected. Dies ist unter Umständen ein Sicherheitsdefizit, welches auch unter dem Namen „Inappropriate Intimacy“ als Code Smell bekannt ist.

#### Verdikt

Das vorliegende Review wurde vom Team akzeptiert.

## Reviews Rails

### Review 1 (08.05.2011)

Review sämtlicher Models

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Beschreibung | Schweregrad | Datum & Kürzel wenn behoben | Req. Ref. | Bemerkungen |
| 1 | Kompliziertes Konstrukt anstelle von Polymorphismus verwendet (Model User => for\_select) | Mittel | 08.05.2011 EL | - |  |
| 2 | Auflösung Model in eine schriftliche Darstellung wird in View gemacht (Duplicated Code) sämtliche Models | Mittel | 08.05.2011 EL | Ja | Extract Method |
| 3 | Es werden Stundeneintragstypen angezeigt, welche nicht mehr gültig sind  (Model TimeEntryType) | Schwer | 09.05.2011 EL | Ja | scope :active |
| 4 | Beim Kunden wird nicht geprüft, ob Nachname angegeben wurde  (Model Customer) | Leicht | 08.05.2011 EL | - | validates :last\_name |
| 5 | Beziehung zwischen Model AppliedMaterial und Material falsch gesetzt. | Mittel | 08.05.2011 EL | - | Model AppliedMaterial has\_many anstatt belongs\_to |

#### Referenz ID 2

Statt einer Methode auf Modelebene (to\_s) um das Model als String darzustellen, wird bei jedem Gebrauch der String mit den passenden Attributen erstellt. Nach Konvention soll eine Methode erzeugt werden, die das Model als String repräsentieren kann (to\_s).

#### Referenz ID 3

Es ist möglich einen Stundeneintragstyp als ungültig zu definieren, jedoch sollten solche Stundeneintragstypen anschliessend bei der Auswahl nicht mehr zur Verfügung stehen. Um dies zu realisieren könnte die Methode scope verwendet werden, welche nur bestimmte Datensätze anzeigt.

#### Verdikt

Das vorliegende Review wurde vom Team akzeptiert.

# Deckung mit der Planung

## Design

## Coderichtlinien

Die Coderichtlinien des Projektes wurden eingehalten.

### Anpassungen

Während der Constructionphase ist die Erfahrung gemacht worden, dass die zuvor festgelegte Spaltenbreite für die von uns eingesetzten Bildschirme zu wenig breit ist. Bei der automatischen Formatierung wurden so Zeilen umgebrochen, obwohl die ganze Zeile ohne Weiteres noch auf derselben Linie Platz gehabt hätte. Die Coderichtlinien wurden dem entsprechend angepasst.