|  |
| --- |
| Elmer Lukas, Heidt Christina, Steiner Diego, Treichler Delia, Waltenspül Remo  31. Mai 2011 |

|  |
| --- |
| SE2 Projekt MRT |
| Qualitätsmassnahmen |
|  |

****

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 04.05.2011 | 1.0 | Erste Version des Dokuments | WR |
| 27.05.2011 | 1.1 | Korrekturen | TD |

## Inhaltsverzeichnis

[1 Dokumentinformationen 1](#_Toc294621135)

[1.1 Änderungsgeschichte 1](#_Toc294621136)

[1.2 Inhaltsverzeichnis 1](#_Toc294621137)

[1.3 Abbildungsverzeichnis 3](#_Toc294621138)

[2 Einführung 4](#_Toc294621139)

[2.1 Zweck 4](#_Toc294621140)

[2.2 Gültigkeitsbereich 4](#_Toc294621141)

[2.3 Definitionen und Abkürzungen 4](#_Toc294621142)

[3 Codestatistiken 5](#_Toc294621143)

[3.1 Android 5](#_Toc294621144)

[3.2 Rails 6](#_Toc294621145)

[4 Codemetriken 8](#_Toc294621146)

[4.1 Android 8](#_Toc294621147)

[4.2 Rails 8](#_Toc294621148)

[5 Codereview 9](#_Toc294621149)

[5.1 Ablauf 9](#_Toc294621150)

[5.2 Fokus bei Review 9](#_Toc294621151)

[5.3 Eingesetzte Tools 9](#_Toc294621152)

[5.3.1 Findbugs 9](#_Toc294621153)

[5.3.2 Checkstyle 9](#_Toc294621154)

[5.3.3 State Of Flow Eclipse Metrics Plugin 9](#_Toc294621155)

[5.4 Reviews Android 10](#_Toc294621156)

[5.4.1 Review 1 (04.05.2011) 10](#_Toc294621157)

[5.4.2 Review 2 (24.05.2011) 10](#_Toc294621158)

[5.5 Reviews Rails 12](#_Toc294621159)

[5.5.1 Review 1 (08.05.2011) 12](#_Toc294621160)

[6 Deckung mit der Planung 13](#_Toc294621161)

[6.1 Design 13](#_Toc294621162)

[6.2 Coderichtlinien 13](#_Toc294621163)

[6.2.1 Anpassungen 13](#_Toc294621164)

## Abbildungsverzeichnis

[Abbildung 1 - Übersicht Codestatistik Android 4](#_Toc294539962)

[Abbildung 2 - Codestatistik Android 4](#_Toc294539963)

[Abbildung 3 - Codestatistik Rails 5](#_Toc294539964)

[Abbildung 4 - Übersicht über die 40 benutzten Libraries 6](#_Toc294539965)

# 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.

# Codestatistiken

## Android

TODO: Sätzli

* Anzahl Packages: 10
* Anzahl eigene Klassen: 60
* Anzahl verwendete Klassen: ~ 35
* Geschriebene Codezeilen: 2517
* Generierte Codezeilen: 66 (R.java)

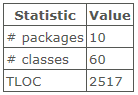


Abbildung 1 - Übersicht Codestatistik Android

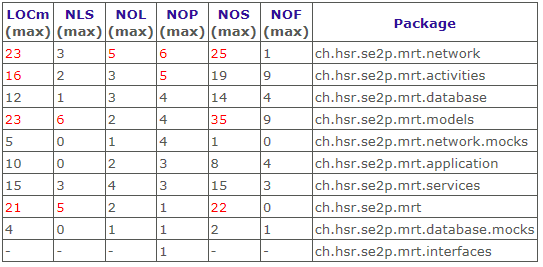


Abbildung 2 - Codestatistik Android

Bildlegende zu Abbildung 2 - Codestatistik Android:

LOCm Lines of Code in Method

NLS Number of Locals in Scope

NOL Number of Levels

NOP Number of Parameters

NOS Number of Statements

NOF Number of Fields

## Rails

TODO: Sätzli

* Anzahl eigene Klassen: 24
* Anzahl verwendete Klassen / Module: ~ 5-20
* Geschriebene Codezeilen
  + Applikation: 1290
  + Test: 1151
  + Views / XHTML: 1111
  + CSS: 464
  + Übersetzungen: 556
  + Total: 4572
* Generierte Codezeilen: ~ 100
* Code / Test Ratio: 1.1/1

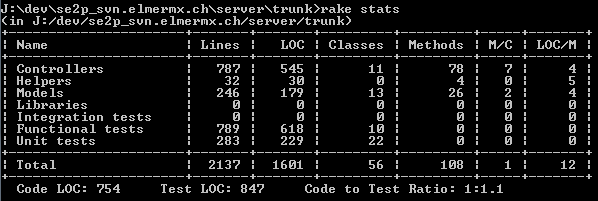


Abbildung 3 - Codestatistik Rails

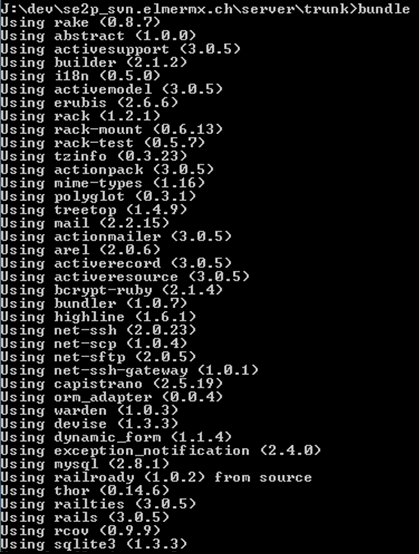


Abbildung 4 - Übersicht über die 40 benutzten Libraries

# Codemetriken

{REMO}

## Android

## Rails

# 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.

### Review 2 (24.05.2011)

Review der Klasse TimeEntryAcivity

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Beschreibung | Schweregrad | Datum & Kürzel wenn behoben | Req. Ref. | Bemerkungen |
| 1 | Klasse zu gross, zusätzliche Klassen für Zeitmessung, LocationService und Spinner erstellen | Hoch | 24.05.2011 EL | Ja | Diverse Refactorings |
| 2 | Zwei verschiedene Spinners vorhanden  -> Attributnamen zu wenig aussagekräftig | Leicht | 24.05.2011 EL | - | Refactoring RenameField |
| 3 | Methodennamen update() zu wenig aussagekräftig | Leicht | 24.05.2011 EL | - | Refactoring RenameMethod |
| 4 | Zurücksetzen der Daten nach dem Stoppen der Zeitmessung in eine eigene Methode extrahieren | Leicht | 24.05.2011 EL | - | Refactoring ExtractMethod |

#### Referenz ID 1

Die Klasse TimeEntryActivity hat zu viele Verantwortlichkeiten. Die Zeitmessung, der Location Service und der Spinner mit der Autocompletion-Funktion sollen je in eine eigene Klasse ausgelagert werden.

#### 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 Construction-Phase 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.