Gerald Bründl

Was ist Software Refactoring?

Als Software Refactoring beschreibt man den Prozess, ein Softwaresystem in seiner internen Struktur zu verbessern ohne dass sich dessen äußeres Verhalten ändert.
Die Ziele des Refactoring sind bessere Lesbarkeit und Verständlichkeit, sowie die Erleichterung der Wart- und Erweiterbarkeit.

Um hinweise auf Stellen im Code zu bekommen die von einem Refactoring profitieren könnten verwendet man einerseits verschiedene Software Metriken, die versuchen die Güte des Codes an spezifischen Maßzahlen festzumachen. Andererseits wird nach Code Smells gesucht, die Codestellen aufzeigen die grundsätzlich schon wie gewünscht arbeiten aber noch in irgendeiner Form schlecht strukturiert sind oder andersartige Mängel aufweisen.

Was macht EMF Refactor?

Der EMF Refactor wendet diese Techniken des Refactorings nun auf Modelle an und erlaubt somit schon Überprüfungen und Refactorings vor dem Beginn der eigentlichen Implementierung durchzuführen. Dadurch können potentiell kritische Designfehler die sonst erst später im Entwicklungsablauf auffallen würden noch problemlos korrigiert werden.

Die Metriken und Smells beziehen sich hierbei nun auf die Eigenschaften und Beziehungen der betrachteten Modelle.

Das Ziel des Modell Refactoring sollen verbesserte Modelle sein die möglichst alle der folgenden Qualitätsmerkmale aufweisen:

-Einfach zu ändern

-Vollständig

-Leicht verständlich

-Angemessen modelliert

-Konsistent

-Korrekt

Verwendung von EMF Refactor

Der EMF Refactor unterstützt das berechnen von Modell Metriken, das auffinden von Modell Smells und die Anwendung von Modell Refactorings.

Jedes der Tools des EMF Refactor basiert auf dem Eclipse Modeling Framework und kann für alle Modelle verwendet werden deren Metamodel Instanzen von EMF Ecore sind, also auch UML 2 Modelle (bisher Klassen und Anwendungsfalldiagramm).

Die Architektur von EMF Refactor ist so aufgebaut, dass es für jedes der drei Features (Metriken generieren, Smells finden, Refactoring) je ein Specification Modul und ein Application Modul gibt.

Das Specification Module unterstützt Java, ocl, Henshin, CoMReL.
Hier werden neue Modell Metriken und Smells und Refactorings generiert.

Das Application Module besteht aus der Configuration Component und der Runtime Component.
Die Runtime Component benutzt den im Specification Modul generierten Java Code und führt die Metrikberechnung, Smell Erkennung oder das Refactoring dann aus.

Will man für ein vorliegendes Modell ein Refactoring durchführen so bietet es sich zuerst an verschiedene Metriken zu berechnen (die frei gewählt werden können) um einen Überblick über die grundlegende Eigenschaften des Modells zu bekommen.

Um spezifische Stellen oder Beziehungen im Modell zu lokalisieren die man möglicherweise besser strukturieren könnte gibt es dann die Möglichkeit das Modell auf Smells zu untersuchen.
Hierbei kann der Benutzer eigene Grenzwerte festlegen und damit die Smells auf die individuellen Anforderungen seines Modells anpassen.
Gefundene Smells und deren Beziehungen untereinander hebt der Papyrus Editor im Modell graphisch hervor.

Nun kann der Benutzer entscheiden welche der gefundenen Smells er durch Refactoring des Modells beheben will. Dazu kann er auch einfach aus den Vorschlägen die zusammen mit den Smells generiert wurden auswählen und diese anwenden.
Der Nutzer muss nur noch die vom jeweiligen Refactoring abhängigen Parameter eingeben.
Falls bei den Eingaben keine Inkonsistenzen bemerkt wurden wird dem Benutzer vor der Durchführung des Refactorings nun mittels EMF Compare eine Vorschau des neuen Modells geliefert, sowie eine ausführliche Analyse der Metriken und Smells vor und nach dem Refactoring.
Dies ermöglicht ein besseres überblicken der Vor- und Nachteile bei jedem Refactoring Schritt.
Somit wird verhindert das eine Verbesserung eines Aspekts des Modells mehrere Verschlechterungen an anderen Stellen nach sich zieht, ohne dass dem Benutzer das bewusst wird.
Wenn der Nutzer nach diesem Schritt immer noch der Meinung ist, dass das Refactoring hilfreich ist, kann er die Modellveränderungen bestätigen und erhält dann ein überarbeitetes Modell.

Es ist für den Benutzer möglich die bereits bestehenden Metriken, Smells und Refactorings über ein Wizard Interface durch selbst entwickelte zu erweitern.
Neue Metriken und Smells können durch Java Code oder im Falle von bestimmten Mustererkennungen auch in Henshin spezifiziert werden.
Eigene Refactorings können ebenfalls in Java oder Henshin implementiert werden.

Aufgabenbeschreibung für das Softwarepraktikum:

Unsere Aufgabe ist es EMF Refactor für weitere UML Diagrammtypen nutzbar zu machen.
Wir müssen also die bestehenden Smells und Metriken und Refactorings in neuen Specification Modulen an die neuen Diagrammtypen anpassen, indem wir die vier Interfaces ImetricCalculateClass, ImodelSmellFinderClass, Icontroller und IGuiHandler implementieren.
Zusätzlich muss auch noch das jeweils dazugehörige Application Module erstellt werden welches mit LTK, BIRT und EMF Compare zusammenarbeiten soll.