Einführung in Software Engineering WS 10/11

Fachbereich Informatik

Dr. Michael Eichberg
eichberg@informatik.tu-darmstadt.de

Assistent: Ralf Mitschke
mitschke@st.informatik.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Übungsblatt 8 (10 Punkte): GRASP, Kopplung und Kohäsion

Abgabeformat: Reichen Sie ihre Lösung per SVN ein. Jede Übung muss in einem eigenen Ordner ***ex<Number>*** *(<Number> = 01, 02, ...)* in Ihrem Gruppenverzeichnis eingereicht werden. Während der Übungsbearbeitung können Sie Ihre Lösungen beliebig oft in das SVN hochladen (per Commit). Wir prüfen die Zeit der Einreichung Ihrer Lösungen unter der Benutzung des SVN Zeitstempels.

Erstellen Sie für Lösungen der Aufgaben, die keinen Quelltext erfordern, eine PDF-Datei mit dem Dateinamen ***solution.pdf***. Dies gilt auch für alle UML-Diagramme, die Sie erstellen. Die Basisanwendung wird als Eclipse-Projekt vorgegeben. Ihr eigener Code muss entsprechend in den dafür vorgesehenen Verzeichnissen (***/src*** oder ***/test***) erstellt werden.

Abgabetermin: 19.01.2011 - 24:00 Uhr

Aufgabe 1 (4 Punkte)

Ziel: *Software Design mit Hilfe von GRASP.*

Verwenden Sie Ihre Flashcards-Anwendung, die Sie bereits in den letzten Übungen erweitert haben. In dieser Aufgabe sollen Sie die Qualität Ihrer bisherigen Version analysieren und darüber hinaus die Strategie für geplante Erweiterungen festlegen.

a) Analyse der bisherigen Implementierung mit Hilfe von GRASP (2 Punkte)

Dokumentieren Sie welche Klassen für die unten aufgelisteten Verantwortlichkeiten zuständig sind. Geben Sie für jedes der drei Prinzipien: Controller, Creator und Information-Expert an, ob es zur Anwendung kam und warum bzw. warum nicht.

- *Die nächste Flashcard während des Lernens auswählen (Achtung: UI ist kein Controller)*
- *Erstellen einer neuen Flashcard (Achtung: UI ist kein Controller)*

b) Erweiterung der bisherigen Implementierung mit Hilfe von GRASP (2 Punkte)

Nutzen Sie nun die selben Prinzipien, um die bisherige Implementierung mit den unten aufgelisteten Verantwortlichkeiten zu erweitern. Listen Sie für jedes der drei Prinzipien incl. Begründung auf, auf welche Klassen es zutrifft. Z.B. alle Klassen, die Information-Expert sind. Lässt sich das Prinzip auf keine Klasse anwenden, begründen Sie warum. Wo würden Sie ggf. die Verantwortlichkeit unterbringen.

- *Zum Vokabellernen sollen Frage und Antwort in allen Flashcards der aktuellen Serie vertauscht werden.*
- *Das automatische Generieren von Flashcards für das kleine Einmaleins, um nicht manuell 100 Flashcards erstellen zu müssen.*

Hinweis: *Verantwortlichkeiten können auch aus mehreren Teilverantwortlichkeiten bestehen.*

Aufgabe 2 (3 Punkte)

Ziel*: Bewerten eines Software-Designs durch Betrachtung von Kopplung.*

Analysieren Sie im Folgenden Ihre eigene Version der Flashcards-Anwendung. Zur automatisierten Unterstützung können Sie das Eclipse-Plugin STAN (http://stan4j.com/general/download-ide.html) nutzen. STAN erlaubt es die strukturellen Abhängigkeiten eines Programmes zu analysieren und zu visualisieren. Mittels STAN können Sie mit Rechtsclick auf das Flashcards-Projekt und durch Auswahl von „Run As...“→„Structural Analysis“ das Projekt analysieren (STAN 2.0.1. ist in einer Community-Version verfügbar, die eine Analyse von bis zu 500 Klassen ermöglicht). Alternativ können Sie die Abhängigkeiten manuell aus dem Quellcode ablesen.

a) Kopplung zu Java Bibliotheken (0,5 Punkte)

Begründen Sie kurz warum bei der Betrachtung von Kopplung das Einbeziehen von Klassen aus der Standard Java Bibliothek nicht sinnvoll ist.

b) Fan-In und Fan-Out (1 Punkt)

Bestimmen Sie den Fan-Out und den Fan-In der Klassen `Flashcard`, `FlashcardSeries` und `FlashcardsWindow`. Testklassen sind hier zu ignorieren.

Hinweis: *Der Fan-Out einer Klasse ist definiert als die Summe aller Klassen zu denen eine Abhängigkeit besteht. Der Fan-In einer Klasse ist die Summe aller Klassen die eine Abhängigkeit zu dieser Klasse haben.*

c) Software-Design und Kopplung (1,5 Punkte)

- Nennen Sie mindestens zwei Nachteile von hoher Kopplung. Beschreiben Sie kurz den Bezug von hoher Kopplung zu den errechneten Fan-In und Fan-Out Metriken. Geben Sie ein Beispiel für hohe Kopplung aus den analysierten Klassen an.
- Was ist die Konsequenz einer häufigen Wiederverwendung einer Klasse (im Hinblick auf die Erweiterbarkeit einer Software)? Beschreiben Sie kurz den Bezug von häufiger Wiederverwendung zu den errechneten Fan-In und Fan-Out Metriken. Geben Sie ein Beispiel für hohe Wiederverwendung aus den analysierten Klassen an.

Aufgabe 3 (3 Punkte)

Ziel: *Bewerten eines Software-Designs durch Betrachtung der Kohäsion.*

Betrachten Sie weiterhin Ihre eigene Version der Flashcards-Anwendung.

a) Kohäsion der Klasse FlashcardsWindow (1,5 Punkte)

Um die Kohäsion K zwischen zwei Methoden M_i und M_j zu definieren, nehmen wir $K = |\sigma(M_i , M_j)|$, wobei $\sigma(M_i, M_j)$ die Menge der von beiden Methoden genutzten Felder angibt. Das heißt $\sigma(M_i , M_j)$ berechnet sich aus den zu M_i, M_j gehörenden Mengen der Variablennutzung V_i , V_j als $\sigma(M_i , M_j) = V_i \cap V_j$.

Zwei Methoden sind dann kohäsiv, wenn $K > 0$ ist.

Stellen Sie eine Liste von Mengen (V) auf, die Ihnen angibt, welche Felder der Klasse `FlashcardsWindow` in welcher Methode genutzt werden (z.B. in der Form: V_{method} = {field1, field2})

Ihre Liste soll alle Methoden, außer den Konstruktoren des `FlashcardsWindow` beinhalten. Sehen Sie für die Variablen-Nutzung alle intern aufgerufene Methoden der gleichen Klasse (sowohl privat als auch öffentlich) als „inlined“ an, d.h. an der Stelle eines Aufrufes an eine private Methode, wird dieser Aufruf vollständig durch die Implementierung der privaten Methode ersetzt. Betrachten Sie daher auch keine Methoden, die Konstruktoren des `FlashcardsWindow` direkt oder indirekt aufrufen.

Nennen Sie ausgehend von den aufgestellten Mengen zwei kohäsive Methoden und zwei nicht kohäsive Methoden der Klasse `FlashcardsWindow`.

b) LCOM* der Klasse FlashcardsWindow (1,5 Punkte)

Nutzen Sie die in der Übungspräsentation vorgestellte Metrik LCOM* und berechnen Sie mit dieser Metrik einen Gesamtwert für die Kohäsion der Klasse `FlashcardsWindow`. Ziehen Sie alle Methoden die in Teilaufgabe a) betrachtet wurden zur Berechnung der Metrik hinzu (Konstruktoren und Methoden, die diese Aufrufen sind weiter zu ignorieren). Behandeln Sie intern aufgerufene Methoden wie in Aufgabe a), d.h. Sie können die in Aufgabe a) aufgestellten Mengen zugrunde legen, um die Parameter der Metrik zu berechnen. Legen Sie nur die Felder zugrunde, die auch von mindestens einer der betrachteten Methoden genutzt wird (Felder die nur in Konstruktoren genutzt werden sind zu ignorieren).

$$\mathrm{LCOM}^{*} = \frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)\right) - m}{1 - m}$$

- a = Anzahl der Felder
- m = Anzahl der Untersuchten Methoden
- $\mu(A_j)$ = Anzahl der Methoden die auf das Feld j zugreifen.

Errechnen Sie die Metrik LCOM* für die Klasse `FlashcardsWindow`. Geben Sie den Rechenweg (a, m, $\mu(A_j)$) in Ihrer Lösung mit an.

Beschreiben Sie kurz wie Sie diesen Wert für die Kohäsion der Klasse `FlashcardsWindow` interpretieren.