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 6 (10 Punkte): Interaktionsdiagramme

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: 09.12.2010 - 24:00 Uhr

Aufgabe 1 (3 Punkte)

Ziel: Analyse des Systemverhaltens mit Hilfe von Systemsequenzdiagrammen

Zur weiteren Verfeinerung des Domänenmodells soll nun eine Analyse des Systemverhaltens durchgeführt werden. Um die definierten Abläufe zu dokumentieren, wurden bereits Ablaufbeschreibungen erfasst. Erstellen Sie ein Systemsequenzdiagramm zu der gegebenen Ablaufbeschreibung des Lernens einer Lernkartei mit der Flashcards-Anwendung.

Lernen einer Lernkartei – Ablaufbeschreibung

Der Benutzer startet den Lernprozess in der Flashcards-Anwendung. In der Anwendung öffnet sich ein Lerndialog, in dem der Benutzer nacheinander seine Karteikarten lernt. Der Lernprozess soll in einem Durchlauf alle im System vorhanden Karten abfragen. Die abgefragten Karten werden in einer zufälligen Reihenfolge gelernt. Hierzu wird innerhalb des Systems jeweils eine zufällige, noch nicht gelernte, Karte ermittelt. Dann präsentiert die Flashcards-Anwendung dem Benutzer die Frageseite dieser Karteikarte. Dies geschieht ebenfalls innerhalb des Lerndialogs. Der Benutzer überlegt sich eine Antwort auf die Frage der Karteikarte. Anschließend wendet er die Karteikarte, um seine Antwort auf Korrektheit zu überprüfen. Die Antwort der Karteikarte wird dann statt der Frageseite innerhalb des Lerndialogs angezeigt. Fällt dem Benutzer während des Lernens ein Fehler auf der Frage- oder Antwortseite der Karteikarte auf, hat er nun die Möglichkeit die Karteikarte zu bearbeiten und anschließend seine Änderungen abzuspeichern. Nach dem Bearbeiten einer Karteikarte wird der Lerndialog aktualisiert, so dass dem Benutzer die geänderte Antwortseite angezeigt wird. Unmittelbar nach dem Vergleich der vermuteten Antwort mit der Antwortseite der Karteikarte, bewertet der Benutzer seine Antwort. Hierzu kann er innerhalb des Lerndialogs auswählen, ob die Antwort positiv oder negativ war. Nach der Bewertung fährt das System sofort mit dem Lernen der nächsten Karteikarte fort, um einen möglichst optimalen Lernfluss zu erreichen. Mit den angegebenen Antwortbewertungen des Benutzers errechnet die Flashcards-Anwendung ein Verhältnis zwischen korrekt beantworteten und der Anzahl der in diesem Durchgang gelernten Karteikarten. Die errechnete Rate wird dem Benutzer direkt nach dem Ende des Lernenprozesses angezeigt.

Aufgabe 2 (7 Punkte)

Ziel: *Darstellung komplexer Abläufe mit Hilfe eines Sequenz-Diagramms*

Im Folgenden soll die Kommunikation innerhalb des AWT-Frameworks (des JDK 1.6) verdeutlicht werden, die stattfindet, um beim Drücken eines **JButtons** die gewünschte Operation der konkreten Anwendung aufzurufen.

*Die AWT-Klasse `JButton` unterscheidet drei Arten von Events: Action-Events, Change-Events und Item-Events. Interessenten für eine bestimmte Art von Event implementieren entsprechend eines der Interfaces **`ActionListener`**, **`ChangeListener`** oder **`ItemListener`** und registrieren sich bei einer Instanz von `JButton`. Bis zum Aufruf der eigenen registrierten Methode(n) findet ein komplexer Prozess innerhalb des AWT-Frameworks statt. Beim Klicken der Maus findet bereit in AWT eine Abbildung der Mauskoordinaten auf AWT-Komponenten statt. Diese Abbildung bestimmt welche der Komponenten tatsächlich eine Nachricht durch AWT erhalten, da sie auf dem entsprechenden Bildschirmbereich der Mausinteraktion liegen. Diese Komponenten werden dann durch AWT darüber benachrichtigt, welche Art von Interaktion stattfand. Im Falle eines einfachen Mausklicks werden zwei Maus-Events („Maus gedrückt“, „Maus losgelassen“) ausgelöst.*

Erstellen Sie **ein** Sequenz-Diagramm das für einen beliebigen `JButton` aufzeigt, welche Events des Buttons (Action-, Change- oder Item-Events) im Laufe eines Mausklicks durch den Button ausgelöst werden. Ein Mausklick besteht aus zwei Maus-Events vom Typ `MouseEvent` mit entsprechenden Identifier: `id = MouseEvent.MOUSE_PRESSED` bzw. `id = MouseEvent.MOUSE_RELEASED`. Starten Sie Ihr Diagramm mit dem Aufruf an die Methode **`mousePressed(MouseEvent)`**, bis der Kontrollfluss zu dieser Methode zurückkehrt, und modellieren Sie anschließend im gleichen Diagramm den Aufruf an **`mouseReleased(MouseEvent)`**. Beide Methoden finden Sie in der JDK 1.6 Klasse **`javax.swing.plaf.baisc.BasicButtonListener`**. Mit dem Aufruf dieser Methoden, durch das AWT-Framework, wurde bereits eine Fallunterscheidung bzgl. der Art des Maus-Events gemacht. Ihr Sequenz-Diagramm soll das wesentliche Szenario, welches zum auslösen der verschiedenen Events von `JButton` (Action-, Change- oder Item-Events) führt, aufzeigen. Für jede Art von Event wird in einem `Button` letztenendes eine zur Art des Events passende Methode aufgerufen, die alle registrierten Listener benachrichtigt:
`fireActionPerformed(ActionEvent)`, `fireStateChanged()`, oder `fireItemStateChanged(ItemEvent)`. Nur für die Methode `fireActionPerformed(ActionEvent)` soll die Interaktion mit registrierten Instanzen von `ActionListener` aufgezeigt werden; für die anderen Events müssen Aufrufe über `fireItemStateChanged(ItemEvent)` und `fireStateChanged()` hinaus nicht weiter verfolgt werden.

Für das Sequenz-Diagramm sollen Sie einige Vereinfachungen vornehmen: Konditionale, deren Blöcke (If, oder Else) in diesem Szenario nicht ausgeführt werden, müssen nicht modelliert werden, um dem Leser des Diagramms die wesentliche Funktion auf einfache Weise zu verdeutlichen. Konditionale, deren Auswertungsergebnis bereits bekannt ist, führen nur den entsprechenden Block aus. Desweiteren sollen nur Methodenaufrufe und Objekte modelliert werden, die zum Pfad der Aufrufe an Event-Listener beitragen. Daher sollte Ihr Diagramm sich auf Instanzen der folgenden Typen beschränken (bzw. deren Implementierungen, oder Methoden aus Superklassen, oder enthaltene innere, und enthaltene anonyme Klassen): `JButton`, `MouseEvent` `BasicButtonListener`, `ButtonModel`, `ActionListener`. Werte, die durch Aufrufe an andere Klassen errechnet werden, können als gegeben vorausgesetzt werden und müssen nicht modelliert werden. Sie können für dieses Szenario davon ausgehen, dass die Maus zwischen den Klicks nicht bewegt wurde und dass es sich um einen Klick mit der linken Maustaste handelte. Desweiteren können Sie davon ausgehen, dass der Button den Zustand ‚enabled‘ hat. **Sie können weitere Vereinfachungen für Ihr Diagramm treffen, sofern diese nachvollziehbar dokumentieren sind!**

Hinweis 1 (Verdeutlichung des `ActionListeners`): *In der Flashcards-Anwendung wird für den Lerndialog der `playButton` erstellt. Durch das Registrieren einer Instanz vom Typ `ActionListener` wird die Anwendung informiert, wenn der entsprechende `JButton` gedrückt wurde. Zur Verdeutlichung der Funktionsweise von AWT bietet es sich an, den Eclipse-Debugger zu verwenden, um die relevanten Aufrufe zu analysieren. Leider können nicht in allen Java-Installationen Breakpoints auf die Methoden `mousePressed(MouseEvent)` und `mouseReleased(MouseEvent)` gesetzt werden. Alternativ, kann z.B. auf dem Aufruf von `learn()` ein Breakpoint gesetzt werden, um durch Analyse des Aufrufstacks herauszufinden, welche Aufrufe und Klassen bis zu diesem Punkt beteiligt waren. Dies bildet allerdings nur einen Teil der geforderten Aufrufe ab.*

```
playButton = Utilities.createToolBarButton(" Learn ",
                "media-playback-start.png", "learn flashcards");
playButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) {
                learn();
        }
});
```

Hinweis 2: *Methoden-Dispatch wird nicht in UML Sequenz-Diagrammen modelliert. Methodenaufrufe auf Interfaces, sind z.B. immer Nachrichten an konkrete Instanzen, einer Klasse die das Interface implementiert.*