Projekt-orientierte Vermittlung von Entwurfsmustern in der Software Engineering Ausbildung

Wilhelm Hasselbring, Jasminka Matevska, Heiko Niemann, Dennis Geesen
Abteilung Software Engineering, Department für Informatik, Universität Oldenburg
{hasselbring|matevska|heiko.niemann}@informatik.uni-oldenburg.de

Hilke Garbe, Stefan Gudenkauf, Steffen Kruse, Claus Möbus
Abteilung Lehr- und Lernsysteme, Department für Informatik, Universität Oldenburg
{hilke.garbe|stefan.gudenkauf}@informatik.uni-oldenburg.de

Marco Grawunder
Software Labor, Department für Informatik, Universität Oldenburg
{marco.grawunder}@informatik.uni-oldenburg.de

Zusammenfassung

Die Projekt-orientierte Ausbildung im Software Engineering ist inzwischen etabliert, z.B. in Form von Softwarepraktika. Im vorliegenden Beitrag berichten wir über unsere Erfahrungen mit der Vermittlung von Entwurfsmustern in diesem Kontext. Wir konzentrieren uns dabei auf die Lehre im Grundstudium und erläutern wie wir hier ein nicht-triviales Softwaresystem Vorlesungs-begleitend als Lehrbeispiel einsetzen. Ergänzt wird die Präsenzlehre durch ein auf Entwurfsmuster spezialisiertes E-Learning- und Assistenzsystem.

1 Einleitung

Ein zentraler Aspekt in der Ausbildung im Software Engineering ist die Vermittlung von Techniken zur Modellierung und der anschließenden programmiertechnischen Umsetzung, beispielsweise in der Kombination von UML und Java [AHM99]. Insbesondere für die Vermittlung von Modellierungstechniken ist es jedoch nicht ausreichend nur die Notationen zu behandeln. Das Erlernen von Software Engineering hat viel damit zu tun, Erfahrungen zu sammeln. Dies kann durch eigenes „Erleben“ z.B. in Form von Übungsaufgaben geschehen, aber auch in der Vermittlung explizit dokumentierten Erfahrungswissens. Ein bewährtes Mittel dazu sind Entwurfsmuster, die bewährte Lösungsstrukturen für häufig wiederkehrende Problemstellungen liefern [GHJV96]. Diese Muster kommen im Allgemeinen erst in Softwaresystemen mit einer

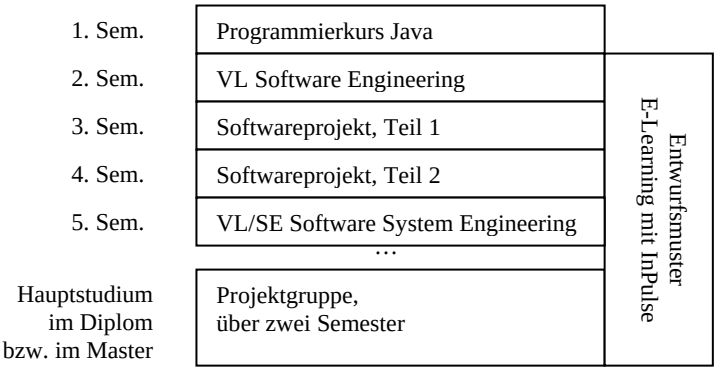
gewissen Komplexität richtig zur Geltung, womit die Studierenden jedoch gerade in den ersten Semestern kaum in Berührung kommen. Ein Ansatz die Studierenden trotzdem mit nicht-trivialen Softwaresystemen zu konfrontieren, ist die Arbeit mit vorgefertigten Beispielen, die die Studierenden (1) verstehen, (2) nachdokumentieren und (3) selbst erweitern sollen. In den ersten zwei Semestern ist es kaum möglich, durch die Studierenden selbst ein komplexes Softwaresystem konstruieren zu lassen. Gleichzeitig ist es sehr schwierig die Notwendigkeit zur Modellierung von Entwürfen zu motivieren, wenn die Studierenden bisher nur kleinere Programme kennengelernt haben (geschweige denn die Notwendigkeit zur Modellierung von Anforderungen zu motivieren, worauf wir in diesem Beitrag jedoch nicht eingehen können). Unser Ansatz besteht nun darin, hierzu ein speziell für die Ausbildung in Software Engineering realisiertes, nicht-triviales Softwaresystem für die Vorlesungs-begleitenden Übungen zu nutzen, im dem insbesondere einige Entwurfsmuster zur Lösung genutzt wurden. Unser Vorgehen besteht hier darin, den Studierenden ein positives Beispiel zu geben. Alternativ könnte auch ein schlecht strukturiertes Beispiel genutzt werden, um die Notwendigkeit zu einer besseren Strukturierung darzustellen, was wir jedoch nicht als geeignetes Motivationsmittel ansehen.

Die Struktur der Lehrmodule im Informatikstudium an der Universität Oldenburg mit direktem Bezug zu Software Engineering ist in Abb. 1 skizziert. Im Programmierkurs werden die Grundlagen der objektorientierten Programmierung mit Java vermittelt [Bol06]. Auf die Vorlesung (VL) Software Engineering, in der Entwurfsmuster eingeführt werden, wird in Abschnitt 2 näher eingegangen sowie auf das zweisemestrige Softwareprojekt in Abschnitt 3. Das als Kombination von Vorlesung und Seminar konzipierte Modul Software System Engineering (Lehrsprache Englisch) vertieft u.a. ausgewählte Muster (beispielsweise für JavaEE) und in der zweisemestrigen Projektgruppe bearbeiten 12 Studierende im Hauptstudium ein Jahr lang ein größeres Projekt. In diesem Beitrag betrachten wir nur die Module des ersten bis vierten Semesters (unabhängig davon, ob es im auslaufenden Diplomstudium oder im seit dem Wintersemester 2000 angebotenen Bachelorstudium stattfindet). Daher gehen wir auch nicht auf die weiteren Module der Praktischen Informatik bzw. auf speziellere Angebote im Software Engineering ein. Das auf Entwurfsmuster spezialisierte E-Learning- und Assistenzsystem InPULSE wird in Abschnitt 4 vorgestellt, bevor wir diesen Beitrag in Abschnitt 5 zusammenfassen und zukünftige Aktivitäten diskutieren.

2 Das Modul Software Engineering

In diesem Vorlesungsmodul werden zunächst Vorgehens- und Prozessmodelle [LL06] sowie das Konfigurationsmanagement behandelt. Das Konfigurationsmanagement wird als grundlegende Technik zur Arbeit im Team bereits so früh eingeführt, damit die Lösungen zu den Vorlesungs-begleitenden Übungen über das Repository abgegeben werden können (wir verwenden Subversion, http://subversion.tigris.org/). Nach einer

Einführung in die Anforderungsanalyse werden Modellierungskonzepte zur Beschreibung von Struktur und Dynamik mittels der UML-Notation erläutert. Auf dieser Grundlage können dann ausgewählte objektorientierte Entwurfsmuster [GHJV96] behandelt werden. Dies geschieht bereits mit Bezug auf das Lehrbeispiel, soweit sinnvoll. Die Vorlesung wird abgeschlossen mit einer Einführung in Techniken zur Qualitätssicherung und zum Projektmanagement. Im Folgenden stellen wir das Lehrbeispiel, die Vorlesungs-begleitenden Aufgaben sowie eine spezielle



Zusatzaufgabe zur Erweiterung des Softwaresystems durch die Studierenden vor.

Abb. 1.: Lehrmodule mit direktem Bezug zu Software Engineering.

2.1 Das Lehrbeispiel File Manager

Im Rahmen der Veranstaltung Software Engineering wird als Lehrbeispiel und Grundlage für vertiefende Übungsaufgaben die Fallstudie „File Manager“ eingesetzt. Diese Fallstudie wurde im Rahmen eines Projekts entwickelt, welches die Schaffung eines praxisnahen didaktischen Lehrkonzeptes zur Ausbildung im Software Engineering zum Thema hatte [Gud04]. Dazu wurden der Softwareentwicklungsprozess und die verschiedenen fachspezifischen Konzepte am Beispiel der Konstruktion eines mittelgroßen Softwaresystems verdeutlicht. Die Entwicklung des File Manager erfolgte mit Hilfe der Entwicklungswerkzeuge Eclipse (www.eclipse.org) und EclipseUML (www.omondo.de).

Abb. 2 stellt einen Screenshot des File Managers dar. Er besteht aus einer Menüleiste (1) und einer Schaltflächenleiste (2), über die sämtliche Funktionalitäten des File Managers zugänglich sind. Zusätzlich können dateispezifische Funktionalitäten über ein Kontextmenü (3) angesprochen werden. Das vom Programm verwaltete Dateisystem wird in einer geteilten Baumansicht (4) visualisiert.

Neben der eigentlichen Implementierung umfasst die Fallstudie u.a. die Ausarbeitung einer exemplarischen Anforderungsdefinition, ein Entwurfsdokument inklusive der verschiedenen Diagramm- und Planhilfen, einen Aufgabenkatalog für Arbeitsblätter und Gruppenübungen, Foliensätze, sowie eine Betrachtung der zum

Einsatz gekommenen Entwurfsmuster und Konzepte. Dabei wurde besonderer Wert auf die Berücksichtigung didaktischer Erkenntnisse gelegt. Insbesondere wurde ein allgemeines Planungsmodell ausgearbeitet, dass die Besonderheiten einer Lehrveranstaltung zum Software Engineering berücksichtigt [Gud04].

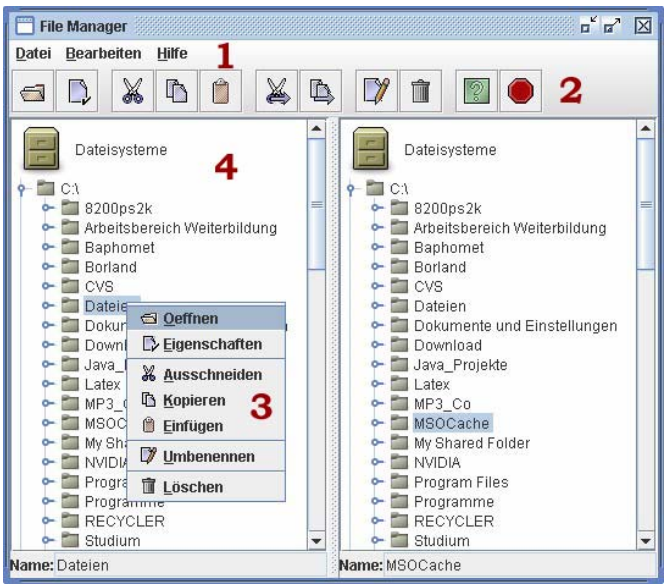


Abb. 2.: Screenshot der Fallstudie File Manager

Die Architektur des File Managers Die Softwarearchitektur des File Managers stellt sich als Zwei-Schichten Architektur dar. Der wesentliche Aspekt ist die Trennung von Zuständigkeiten in Form der Trennung der Programmvisualisierung von der Programmlogik. Der strukturelle Aufbau ist in Abb. 3 dargestellt. Die Architektur ist folgendermaßen aufgebaut:

- Die Benutzungsschnittstelle ist im Paket `presentation` wiederzufinden. Sie besitzt das Unterpaket `swing`, in dem Java Swing Elemente zur grafischen Darstellung gekapselt sind.
- Die Dateilogik ist die Anwendungsebene des Systems. Sie setzt direkt auf dem Datei- und Verzeichnissystem der Zielumgebung auf und stellt Operationen auf diesem bereit. Die Dateilogik ist im Paket `logic` realisiert. Sie besitzt das Unterpaket `file`, das die direkte Datei- und Verzeichnisbehandlung übernimmt.

Dieser Aufbau erlaubt den einfachen Austausch sowohl der Benutzungsschnittstelle als auch der Dateilogik. Eine denkbare alternative Realisierung der Dateilogik wäre z.B. die Verwaltung des Repositories eines Versionsmanagementsystems, wie Subversion. Kern der Trennung von Visualisierung und Logik bildet die Schnittstelle

`EntityInterface`, welche generische Entitäten darstellt, die von der Benutzungsschnittstelle darzustellen und von der Dateilogik zu behandeln sind.

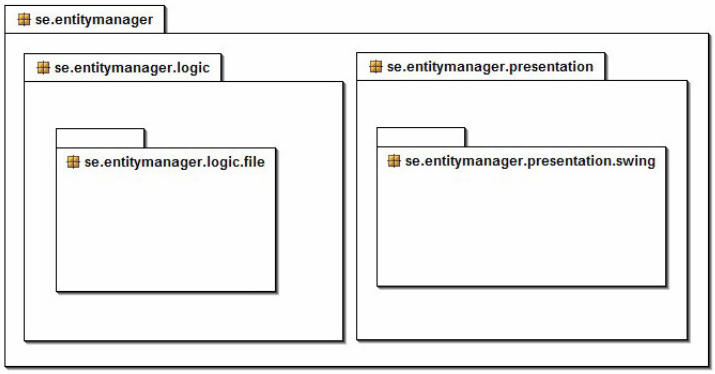


Abb. 3.: UML-Paketdiagramm des File Managers

Entwurfsmuster in der Architektur des File Managers Obwohl die Fallstudie File Manager kein extrem großes System ist, enthält dessen Implementierung zahlreiche Entwurfsmuster zur Anschauung und zur Lehre. Diese sollten u.a. von den Teilnehmern der Veranstaltung Software Engineering im Programmcode entdeckt und beschrieben werden (siehe Abschnitt 2.2). Tabelle 1 stellt eine Auswahl der zu findenden Entwurfsmuster und deren Verwendung vor. Aus Platzgründen können wir in diesem Papier nicht näher auf den Detailentwurf eingehen.

Tab 1.: Auswahl einiger Entwurfsmuster in der Architektur des File Managers

Entwurfsmuster	Verwendung
Facade	Fassaden ermöglichen sehr schmale Kommunikationsschnitt-stellen zwischen Benutzungsoberfläche und Dateilogik und kapseln damit beide Schichten ein.
Singleton	*Singleton*-Muster stellen die einmalige Instanziierung von Klassen sicher und sorgen dafür, dass die *Singleton*-Klassen wohlbekannte Ansprechpunkte für Clients darstellen. *Singleton*s treten im File Manager u.a. im Zusammenspiel mit Fassaden auf.
Abstract Factory	Durch die Bereitstellung einer Schnittstelle, die die Erstellung von Symbolen beschreibt ohne eine konkrete Implementierung zu spezifizieren, und der zugehörigen implementierenden Klassen werden die Austauschbarkeit und die Konfigurierbarkeit der Symbolverwendung innerhalb des File Managers gewährleistet
Factory Method	Mit Hilfe des *Factory Method*-Musters werden die Verzeichnis- und Dateistrukturen repräsentierende Entitäten generiert, ohne dass

	bekannt ist, zu welcher (Datei-)Art diese gehören.
Observer	Die Verwendung des *Observer*-Musters ermöglicht, auf Änderungen der Entitäten automatisch zu reagieren. Ein explizites Aktualisieren der Benutzungsoberfläche ist nicht notwendig.
Composite	Mit Hilfe des *Composite*-Musters wird die Eigenart der Datei-en, andere Dateien enthalten oder nicht enthalten zu können (z.B. Dateiarchive), sowie die unterschiedliche Behandlung von Dateien und Ordnern optimal wiedergegeben.

2.2 Begleitende Aufgaben

Im Modul Software Engineering werden wöchentlich Übungsaufgaben gestellt, die in Tutorien mit ca. 20 Teilnehmern besprochen werden. Die Modellierungsaufgaben werden dabei mit UML-Werkzeugen gelöst: In 2006 wurde Poseidon (www.gentleware.com) eingesetzt, in den Jahren davor Together (www.borland.com). Neben Aufgaben aus unterschiedlichen Anwendungsbereichen werden auch Aufgaben zur Fallstudie File Manager (siehe Abschnitt 2.1) und zu Entwurfsmustern gestellt, die nach folgender Klassifizierung vorgestellt werden sollen:

- Allgemeine Aufgaben zum File Manager (ohne Entwurfsmuster)
- Aufgaben zu Entwurfsmustern
- Aufgaben zu Entwurfsmustern im File Manager

Zunächst werden Anwendungsfälle und Interaktionen hinsichtlich der Fallstudie File Manager modelliert. Hierbei geht es in den Tutorien darum, die studentischen Lösungen mit dem Entwurf des File Managers zu vergleichen, um, wie bereits erwähnt, an einem guten Beispiel zu lernen. Weiterhin wird auch ein Reverse Engineering des File Managers bzw. von Teilen des File Managers vorgenommen.

Bei den „einfachen“ Aufgaben zu Entwurfsmustern wird das benötigte Muster in der Aufgabenstellung vorgegeben. Hierbei wird die Verwendung des Entwurfsmusters in einer konkreten Problemstellung geübt, z.B. wurde das Entwurfsmuster Singleton bei der Implementierung einer Komponente für eindeutige Schlüsselgenerierung oder das Entwurfsmuster Proxy bei der Modellierung der Stellvertreterrolle eines Managers gegenüber seinen Sportlern genutzt. Bei den „komplexen“ Aufgaben konnte auf das E-Learning-System InPULSE (siehe Abschnitt 4) zurückgegriffen werden. Hiermit kann über das Dialogsystem ein passendes Entwurfsmuster bestimmt werden, z.B. für eine Playlist von Musiktiteln das Entwurfsmuster Flyweight oder für einen Stecker eines Föns, der im Ausland genutzt werden soll, das Entwurfsmuster Adapter. Nach der Bestimmung des Entwurfsmusters kann InPULSE bei der folgenden Modellierung durch hinterlegte Informationen zu den Mustern helfen, was z.B. in einer Aufgabe zur Modellierung einer Plastikartikelproduktion mittels des Entwurfsmusters Abstract Factory geschehen ist. Bei den Aufgaben zu Entwurfsmustern im File Manager wird z.B. auf die Ergebnisse aus der Aufgabe „Reverse Engineering“ (siehe oben) zurückgegrif-fen. Die verwendeten Entwurfsmuster sollen identifiziert und der Zweck

ihres Einsatzes diskutiert werden. In 2006 wurde auch eine Zusatzaufgabe für besonders interessierte Studierenden gestellt, die über den Rahmen der normalen Aufgaben hinwegging. Daher wurde für diese Aufgabe eine vierwöchige Bearbeitungszeit gestattet. Zur Motivation wurde eine Prämie für die beste Lösung ausgelobt. Diese Zusatzaufgabe wird im folgenden Abschnitt dargestellt. Zukünftig könnte dies zur Pflicht werden.

2.3 Zusatzaufgaben zur Erweiterung des File Managers

Die Zusatzaufgabe bestand darin, die Funktionalität des File Managers sinnvoll zu erweitern. Die Erweiterung soll durch den Einsatz weiterer Entwurfsmuster modelliert und implementiert werden. Die Lösung sollte eine entsprechende Dokumentation enthalten. Das Interesse an dieser Aufgabe war recht groß und dabei wurden verschiedene Lösungen vorgestellt. Sie umfassten z.B. die Erweiterung der Funktionalität in Form einer Vorschauansicht (z.B. für verschiedene Bildformate) mit Nutzung des Erbauer (Builder) Musters, Unterstützung zur Behandlung von versteckten Dateien unter Einsatz des Singleton-Musters, Suche nach doppelten Dateien (Memento-Muster), Sortierfunktion mit Hilfe des Decorator-Musters und schließlich eine Erweiterung um einen Undo/Redo-Mechanismus unter Einsatz der Muster Memento und Singleton, die im Folgenden ausführlicher beschrieben wird.

Um die gewünschte Funktionalität eines Undo/Redo-Mechanismus zu realisieren eignet sich das Memento-Muster. Weiterhin sollte es nur eine Instanz des Memento-Musters geben, damit sichergestellt wird, dass das zuletzt gespeicherte Memento eindeutig ist. Deswegen wurde zusätzlich das Singleton-Muster verwendet.

Das Memento-Muster dient im Allgemeinen dazu, einen Zustand eines Objekts zu speichern, so dass es nach einer Änderung wieder in den alten Zustand zurückversetzt werden kann. Das Muster besteht aus einem Urheber, einem Aufbewahrer und einem oder mehreren Mementos. Der Urheber verwaltet die Mementos, erzeugt diese und gibt diese weiter nach außen. Der Aufbewahrer fordert das Memento vom Urheber an und kommuniziert nur über den Urheber mit den Mementos. Der Aufbewahrer enthält ebenso alle Mementos. Das Singleton-Muster ist anzuwenden, wenn man sicherstellen will, dass es nur eine einzige Instanz einer Klasse geben darf. Ein von außerhalb geschützter Konstruktor und eine statische Methode, die die einzige Instanz wiedergibt, stellt diese Funktionalität bereit. Für die Verwendung des Memento-Musters musste dieses den Anforderungen der Erweiterung angepasst werden. Da es aufgrund der verschiedenen Aktionen wie Kopieren, Umbenennen oder Löschen und deren unterschiedlichen Anforderungen an das Memento auch mehrere unterschiedliche Mementos geben muss, dient hier das Memento als abstraktes Interface. Der Entwurf der Erweiterung ist in Abb. 4 dargestellt.

Die speziellen Mementos implementieren das Interface Memento, damit diese vom Aufbewahrer, der Klasse MementoManager, gespeichert werden können. Als Datenstruktur wurde hier ein Keller (Stack) vom Datentyp Memento gewählt, da diese Struktur die chronologische Verwaltung der Mementos vereinfacht. Weiterhin wurden

zwei Actions hinzugefügt, eine UndoAction und eine RedoAction, die je dafür Sorge tragen, eventuelle Undo- oder Redo-Mementos wieder herzustellen. Diese Actions gehören im Vergleich zu der Mustervorlage zu dem Urheber, da nur der Urheber direkt auf die Mementos zugreifen darf. Diese Actions setzen und erzeugen die Mementos. Weiterhin darf es nur eine Instanz des MementoManagers (Aufbewahrer) geben, um den Kontrollfluss über die Mementos eindeutig zu halten. Daher wird der MementoManager mit Hilfe des Singleton-Musters erstellt. Diese Vorgehensweise hat ebenfalls den Vorteil, dass die Klassen UndoAction und RedoAction immer auf dieselbe Instanz des MementoManagers zugreifen. Den Nutzerinnen und Nutzern des File Managers stellen sich die Mementos als durchgeführte Aktionen dar, die wieder rückgängig gemacht werden können.

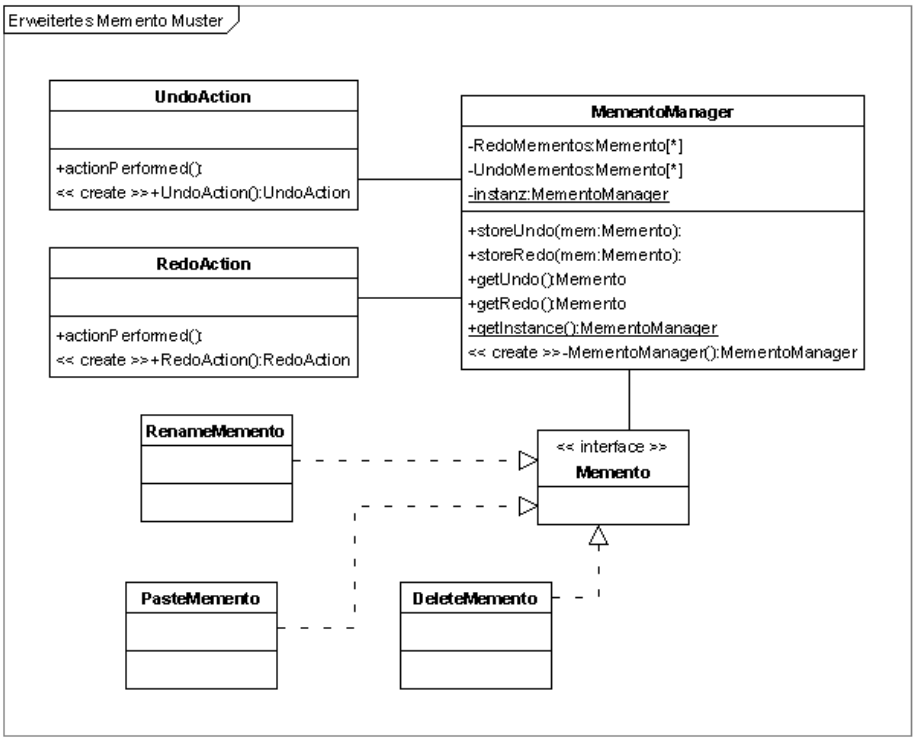


Abb. 4.: Abstraktes, angepasstes Memento- und Singleton-Muster

Das geänderte Muster konnte durch die Studierenden in die durch den File Manager vorgegebenen Struktur integriert werden. Dazu wurde die Logik des File Managers erweitert, indem jede existierende Funktion bei ihrem Aufruf ein entsprechendes Memento erstellt und dieses dem MementoManager übergibt. Ebenso wurde die

grafische Benutzungsoberfläche um zwei Menüeinträge und zwei Schaltflächen für die Undo- und Redo-Funktion erweitert. Weiterhin erfolgten einige Änderungen an der bestehenden Logik, damit die Funktionalität der Mementos sichergestellt ist. Neben den neuen Methoden musste dabei beachtet werden, dass diese auch den vorher gegebenen Entwurfsmustern des File Managers genügen.

3 Das anschließende Softwareprojekt

Das Softwareprojekt schließt sich im SE-Kurrikulum direkt an die Veranstaltung Software Engineering an und führt Wissen aus verschiedenen Grundstudiums-Veranstaltungen zur praktischen Anwendung. Diese Pflichtveranstaltung im Umfang von 4 SWS findet, wie in Abb. 1 dargestellt, im 3. und 4. Semester statt und hat zum Ziel, Techniken der systematischen Softwareentwicklung im Team zu vermitteln. Regelmäßig findet diese Veranstaltung auch standortübergreifend statt [BHASV03].

Im Softwareprojekt entwickeln Gruppen mit einer Stärke von 8-12 Personen ein komplexes softwaretechnisches System, i.d.R. mindestens Client-/Server-basiert. Die Aufgabe ist dabei so komplex gewählt, dass sie nur durch die geschickte Aufteilung in Teilaufgaben effektiv gelöst werden kann. Auf diese Weise ist die Gruppe gezwungen, alle Teilnehmer in der Bearbeitung der Aufgabe zu involvieren. Die Gruppen werden durch einen Tutor unterstützt und durchlaufen alle klassischen Phasen der Softwareentwicklung in einem auf dem Wasserfallmodell basierendem Vorgehensmodell [LL06]. Neben der eigentlichen Softwareentwicklung müssen die Studierenden im ersten Semester basierend auf groben Vorgaben und im zweiten Semester selbstständig auch eine Projektplanung und –überwachung durchführen.

Die Veranstaltung beinhaltet einen Vorlesungsblock, Gruppensitzungen, Präsentationen sowie ein Proseminar. Im Vorlesungsblock zu Beginn des Semesters werden wesentliche, für die Bearbeitung der jeweiligen Aufgabenstellung notwendige Grundlagen, wie die Java-basierte Webentwicklung, vermittelt. Hierbei wird bewusst auf eine Wiederholung softwaretechnischer Grundlagen verzichtet und lediglich ein Handlungsrahmen vorgestellt, der beim eigentlichen Vorgehen helfen soll. Die Gruppen führen mindestens einmal pro Woche zusammen mit einem Tutor eine Sitzung durch, in der wichtige inhaltliche und organisatorische Entscheidungen getroffen, Vorträge zu Themen des Praktikum gehalten und Ergebnisse mit dem Tutor diskutiert werden. Ein weiterer wesentlicher Block ist das Einüben von Präsentationen. Jedes Mitglied der Gruppe kann sich zu Beginn des Semesters eine Spezialaufgabe aus einer Liste vordefinierte Aufgaben (Projektmanagement, Versionsverwaltung, Datenbank, GUI, etc.) aussuchen und muss die Gruppe bezüglich dieses Themas beraten und einen Vortrag zu diesem Thema vorbereiten. Weiterhin findet eine Art Messe etwa nach 2/3 des Projektes statt, in denen die Gruppen ihre bis dahin vorliegenden Ergebnisse einem größeren Publikum im Rahmen des Hochschulinformationsstages vorstellen. Jeweils zum Abschluss eines Semesters erfolgt eine Abnahme des entwickelten Produkts.

Generell ergeben sich beim Softwareprojekt für die Studierenden Probleme, die sich im Wesentlichen auf mangelnde Erfahrung bei der Software Entwicklung zurückführen lassen. So haben sie häufig Probleme, die komplexe Aufgabe in kleinere Teilaufgaben herunterzubrechen und entsprechende Abstraktionen durchzuführen. Es fehlt ein Gefühl dafür, was ein guter Entwurf ist. Kopplung und Kohäsion scheinen eine zu abstrakte Metrik zu sein. Eine Fassade war praktisch ein Fremdwort. Gerade bei diesen Problemen hat sich in den letzten beiden Durchläufen des Softwareprojekts eine entscheidende Verbesserung ergeben. Durch die stärkere Konzentrierung auf den Aspekt der Entwurfsmuster in der Software Engineering Veranstaltung fällt es den Studierenden deutlich leichter, eine sinnvolle Aufteilung und Kapselung von Modulen durchzuführen, was insgesamt zu deutlich besseren Entwürfen und einer effektiveren Untergliederung der Aufgabe führte. Das im nächsten Abschnitt vorgestellte InPULSE-System kann von den Studierenden kontinuierlich zum Selbststudium und zur Vertiefung genutzt werden.

4 Das E-Learning- und Assistenzsystem InPULSE

Das E-Learning- und Assistenzsystem InPULSE (http://inpulse.uni-oldenburg.de) wurde in den vom BMBF geförderten Projekten estat [MAHTZ02] und InPULSE entwickelt. Es dient zum einen als E-Learning-System, in dem Informationen zu Entwurfsmustern abrufbar sind. Zum anderen enthält es ein Assistenzsystem, welches einen Dialog mit den Benutzern über ihr aktuelles Entwurfsvorhaben führen und anschließend Vorschläge zum Einsatz von Entwurfsmustern geben kann [GJMSV06]. Während des Dialogs werden den Benutzern sukzessiv Fragen zu ihrem Entwurf gestellt, um ihre Modellierungssituation zu erfassen. Abb. 5 zeigt exemplarisch eine Seite aus dem System. Im linken Bildschirmbereich sind die vorhandenen Muster über eine Baumdarstellung zugriffsbereit. Im rechten Bereich wird über die Karteikarten-Reiter-Darstellung ein schneller Zugriff auf die einzelnen Elemente ermöglicht, die in allen Musterbeschreibungen vorhanden sind. Die Elemente sind der Beschreibung der Entwurfsmuster nach [GHJV96] angelehnt. Besonderer Wert wurde bei den Inhalten darauf gelegt, dass sie für Software Engineering Anfänger geeignet und verständlich sind, dies gilt insbesondere für die enthaltenen Beispiel-Anwendungen für die Muster. Um die Anwendung der Muster zu unterstützen, besteht die Möglichkeit, die Struktur des Musters und des Beispiels als XMI-Dateien im System zu hinterlegen. So können die Nutzer diese Diagramme in die von ihnen verwendeten CASE-Tools importieren und als Vorlage für ihren Entwurf verwenden. Ebenso können die Quelltexte der Beispiele als Java-Dateien heruntergeladen werden, die mit wenig Aufwand zu ablauffähigen Programmen vervollständigt werden können. Bei einer durchgeführten Evaluation des InPULSE-Systems mit Studierenden der Vorlesung Software Engineering wurden die Download-Möglichkeiten sowohl der Diagramme als auch des Quelltextes als positiv bewertet: „Die Muster sind gut erklärt/ vorgestellt, vor allem, da

Diagramme und Beispielcode zur Verfügung stehen“ und „Schön, übersichtlich und gute Beispiele sowie Beispielcode“.

Weiterhin bietet InPULSE die Möglichkeit, aus vorhandenen Elementen Kurse zusammenzustellen, z.B. speziell für unser Fallbeispiel. Lehrende können den Umfang und die Reihenfolge der Elemente festlegen, die von den Studierenden bearbeitet werden sollen. Um eine Adaption an die Lehrveranstaltung vorzunehmen, können diese Elemente im Kurs durch eigene veranstaltungsspezifische Elemente ergänzt werden. Z.B. kann die Einbettung der einzelnen Muster in ein größeres Softwaresystem als Fallstudie durch zusätzliche Kurs-Elemente gezeigt werden. So kann InPULSE zur gezielten Unterstützung einer Veranstaltung genutzt werden oder als freies Nachschlagewerk Projektarbeiten von Studierenden unterstützen.

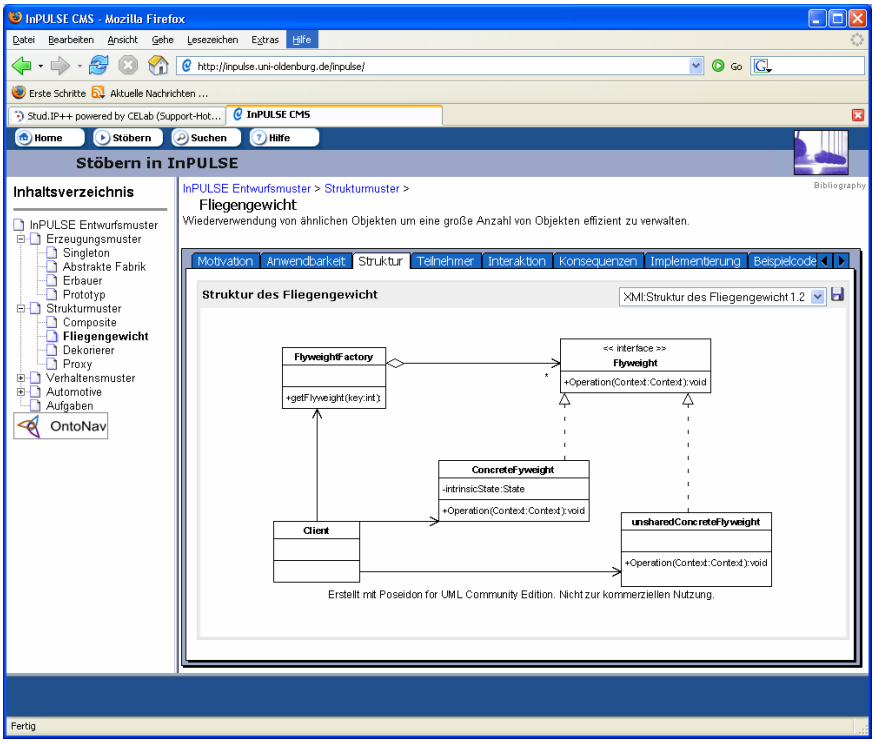


Abb. 5.: Screenshot von InPULSE.

5 Schlussbetrachtung

Die Projekt-orientierte Vermittlung von Entwurfswissen in Form von Entwurfsmustern, die in einem nicht-trivialen Beispielsystem umgesetzt und durch die Studierenden nachdokumentiert sowie erweitert werden, hat sich in der Informatikausbildung an der Universität Oldenburg bewährt. Insbesondere im Softwareprojekt und in

Projektgruppen konnte festgestellt werden, dass die Studierenden über mehr Entwurfswissen verfügen, als dies zuvor der Fall war. Die Nutzung von Entwurfsmustern ist für die Studierenden zu einer Selbstverständlichkeit geworden.

Inzwischen haben wir den File Manager öffentlich über Sourgeforge zugänglich gemacht, so kann zukünftig in den Aufgaben im Lehrmodul Software Engineering auf http://filemanager-uol.sourceforge.net/ verwiesen werden. Dies ermöglicht es den Studierenden, ihre Erweiterungen auch dort zu veröffentlichen und so Erfahrungen mit der Arbeit an Softwaresystemen in einem größeren Kontext zu gewinnen.

Der Inhalt in InPULSE kann für spezielle Veranstaltungen angepasst werden. Die Aufnahme umfassenden Kursmaterials zum File Manager in das InPULSE-Kurssystem kann das vorhandene Lehrangebot abrunden. Ein solcher Kurs würde die für den File Manager relevanten Entwurfsmuster einführen, deren Verwendung in einer in sich abgeschlossenen Anwendung aufzeigen und einen beispielhaften Anwendungskontext für das Dialogsystem bieten. Weiterhin werden in InPULSE neben den klassischen objektorientierten Entwurfsmustern [GHJV96] zur Zeit auch Muster auf Architekturebene [Has06] für das E-Learning aufbereitet. Wir nehmen an, dass die Nutzung des Dialogsystems das Lernen der Entwurfsmuster unterstützt, da das System gezielt entwurfsrelevante Aspekte der Muster abfragt und so zur Reflexion über diese Aspekte und den eigenen Entwurf anregt. Für das nächste Jahr ist eine entsprechende Evaluation vorgesehen.

6 Literaturverzeichnis

[AHM99] K. Alfert, W. Hasselbring, A. Mester: Von Analyse und Entwurf zur Programmierung in einem Semester: UML, Java und deren Anwendung in Projektteams. In: Software Engineering im Unterricht der Hochschulen SEUH'99. Teubner 1999.

[BHASV03] L. Bischofs, W. Hasselbring, H.-J. Appelrath, J. Sauer, O. Vornberger: Erste Erfahrungen mit dem Virtuellen Softwareprojekt. In: Software Engineering im Unterricht der Hochschulen SEUH'8. dpunkt 2003.

[Bol06] D. Boles: Programmieren spielend gelernt mit dem Java-Hamster-Modell, 3. Auflage, Teubner, 2006

[GHJV96] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, 1996

[GJMSV06] H.Garbe, C. Janssen, C. Möbus, H. Seebold, H. de Vries: KARaCAs: Knowledge Acquisition with Repertory Grids and Formal Concept Analysis for Dialog System Construction. In: 15th International Conference on Knowledge Engineering and Knowledge Management - Managing Knowledge in a World of Networks (EKAW 2006), to appear 2006.

[Gud04] S. Gudenkauf: Entwicklung eines Anwendungsbeispiels für die Ausbildung im Software Engineering mittels EclipseUML. Individuelles Projekt, Carl von Ossietzky Universität Oldenburg, Department für Informatik, Abteilung Software Engineering. 2004.

[Has06] W. Hasselbring: Software-Architektur. *Informatik-Spektrum*, 29(1): 48–52, Februar 2006.

[LL06] J. Ludewig, H. Lichter: Software Engineering. dpunkt 2006.

[MAHTZ02] C. Möbus, B. Albers, S. Hartmann, H.J. Thole, J. Zurborg: Towards a Specification of Distributed and Intelligent Web Based Training Systems. In: Intelligent Tutoring Systems, Proceedings of the 6th International Conference (ITS2002), Biarritz, France and San Sebastian, Spain, June, 2002, S. 291–300, Berlin: Springer, Lecture Notes in Computer Science, LNCS 2363