Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

Eine Werkstatt zum Vermitteln objektorientierter Entwurfs- und Sprachkonzepte mit Teachlets

Axel Schmolitzky
Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg
schmolitzky@informatik.uni-hamburg.de

Zusammenfassung

Teachlets sind eine neuartige Lehrform, die ursprünglich für die Vermittlung von Entwurfsmustern entwickelt wurde. Ausgehend von lauffähigem Code wird eine Aufgabe gestellt, die gemeinsam und interaktiv von allen Teilnehmern innerhalb einer Unterrichtseinheit gelöst wird, ein Moderator bedient dabei auf Zuruf den Rechner mit Entwicklungsumgebung und Beamer. Teachlets können bereits für sich genommen als innovative Lehrform eingesetzt werden, sie können aber auch als Entwurfsgegenstand in seminarartigen Veranstaltungen dienen. In der hier beschriebenen Lehrveranstaltung zu fortgeschrittenen Konzepten objektorientierter Programmierung haben die Teilnehmer Teachlets in einer Teachlet-Werkstatt selbst entwickelt und eingesetzt.

1 Einführung

Entwurfsmuster (siehe unter anderem [GHJV95]) sind eine interessante Herausforderung in der Informatik-Ausbildung. Der Raum der Möglichkeiten zu ihrer Vermittlung kann aufgespannt werden von einer reinen Vorlesung ohne Übungen bis hin zu lose betreuten Aufgabenstellungen, die den Einsatz von Entwurfsmustern nahe legen. Da Entwurfsmuster eng mit praktischen Entwurfserfahrungen verknüpft sind, birgt eine reine Vorlesung die Gefahr, dass lediglich Schlagworte hängen bleiben. Immerhin kann in einer Vorlesung eine breite Abdeckung der wichtigsten Entwurfsmuster gewährleistet werden. Eine lose Betreuung von Aufgaben mit hohem Praxisanteil hingegen (also ein Praktikum) hat den Vorteil der individuellen praktischen Erfahrung, aber auch Nachteile: Das individuelle Feedback zu einzelnen Entwurfsentscheidungen kommt meist zu kurz, und nicht alle Teilnehmer kommen möglicherweise mit allen relevanten Entwurfsmustern in Kontakt.

Teachlets wählen einen Mittelweg zwischen Frontalvermittlung und individueller Auseinandersetzung, indem in einer speziell vorbereiteten Unterrichtseinheit alle Teil-

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

nehmer gemeinsam an einer Aufgabenstellung arbeiten. Der Ansatz trägt unter anderem der nachdrücklichen Aussage von Vlissides in [CHV00] Rechnung, dass Entwurfsmuster immer im Zusammenhang mit den Problemen, die sie lösen, betrachtet werden sollten. Das Konzept der Teachlets wurde ursprünglich zur Vermittlung von Entwurfsmustern entwickelt, scheint aber Potenzial für eine breitere Anwendung zu haben. In diesem Artikel werden primär Teachlets zur Vermittlung von Entwurfsmustern diskutiert, weil hier die meiste Erfahrung gesammelt wurde. Im Ausblick werden weitere mögliche Anwendungsfelder angesprochen.

Das Konzept wurde in einem Projektseminar im Sommersemester 2004 an der Universität Hamburg erstmalig vorgestellt und von den Teilnehmern sehr positiv aufgenommen. Der Veranstalter führte zu Beginn einige Teachlets selbst durch, um das Konzept zu verdeutlichen. Anschließend hat im Laufe der Veranstaltung jeder Teilnehmer mindestens ein Teachlet selbst entwickelt und an der Gruppe ausprobiert.

Dieser Artikel ist in erste Linie ein Erfahrungsbericht. Im folgenden Abschnitt wird zunächst das Grundkonzept eines Teachlets vorgestellt. Abschnitt 2 stellt ein Beispiel dar und Abschnitt 3 fasst die Erfahrungen zusammen, die bisher bei der Durchführung von Teachlets gesammelt wurden. In Abschnitt 4 wird das Konzept einer seminarartigen *Teachlet-Werkstatt* dargestellt, in der die Teilnehmer selbst Teachlets entwerfen und auf diese Weise intensiv mit dem zu vermittelnden Stoff umgehen müssen. Abschnitt 5 gibt einen Ausblick auf weitere Anwendungsfelder sowie auf Aktivitäten, die den Ansatz weiter erkunden können.

2 Teachlets

Die folgende Definition bildet die programmatische Grundlage des Teachlet-Konzepts:

„Ein Teachlet ist eine interaktive Lehreinheit, in der ein lauffähiges Stück Software um eine klar definierte Funktionalität erweitert werden soll, um ein Entwurfsmuster oder ein Programmiersprachkonzept zu veranschaulichen. Ein Moderator motiviert mit Hilfe eines Rechners und eines Beamers das Ausgangssystem sowie die vorzunehmende Erweiterung und lässt sich dann von den Teilnehmern anleiten, die dazu notwendigen Änderungen am Quelltext vorzunehmen.“

Die Grundidee eines Teachlets ist somit, dass durch den praktischen Umgang bei den Teilnehmern ein größerer Lerneffekt als mit rein theoretischen Diskussionen erzielt werden soll. Eine Betonung liegt hier auf dem Wort *interaktiv*, denn der Moderator eines Teachlets demonstriert nicht einfach nur eine praktische Realisierung, sondern motiviert die Teilnehmer zur aktiven Gestaltung einer Lösung.

2.1 Statische Struktur eines Teachlets

Die folgende vier Bestandteile sind unabdingbare Voraussetzung für ein erfolgreiches Teachlet und sollten explizit bei der Vorbereitung berücksichtigt werden:

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

- Ein *Ausgangssystem*: eine möglichst kleine, lauffähige Anwendung, die im Quelltext vorliegt.
- Ein *Lernziel*: Kenntnis eines Entwurfsmusters oder eines Programmiersprachenkonzepts, das praktisch kennen gelernt und verstanden wurde.
- Eine oder mehrere *Aufgaben*: Sie implizieren eine Erweiterung/Änderung am Ausgangssystem; beim Lösen der Aufgaben soll das Lernziel durch gemeinsame Programmierung erreicht werden.
- Ein *Foliensatz*: zur Darstellung von Ausgangssystem, Lernziel und Aufgabe, zur Erläuterung grundlegender Terminologie, aber auch zur Diskussion von Lösungsalternativen und Verallgemeinerungen; ergänzend sind Hand-outs möglich.

Das Ausgangssystem sollte klein genug sein, um von den Teilnehmern in 15 bis 20 Minuten erkundet werden zu können. Dabei sollte deutlich zwischen (nach außen sichtbarer) Funktionalität und (innerer) Architektur unterschieden werden. Die Funktionalität sollte klar beschrieben bzw. leicht erkundbar sein. Als sehr motivierend haben sich Ausgangssysteme mit einer grafischen Oberfläche erwiesen.

2.2 Dynamik eines Teachlets

Die Qualität eines Teachlets zeigt sich erst bei der Durchführung in einer Unterrichtseinheit. Abhängig von den beteiligten Personen können, ausgehend von derselben statischen Teachlet-Struktur, unterschiedliche Verläufe auftreten. Ein Einsatz eines Teachlets in einer Lehr- oder besser Lerneinheit wird im Folgenden als *Teachlet-Einheit* bezeichnet. Eine Teachlet-Einheit verhält sich zur statischen Beschreibung aus dem vorigen Abschnitt wie ein Exemplar zu seiner Klasse in der Objektorientierung. Eine typische Teachlet-Einheit dauert 90 Minuten.

Jede Teachlet-Einheit hat eine Anzahl an *Teilnehmern* und einen *Moderator*, der das Teachlet mit Hilfe eines Rechners (dem *Teachlet-Rechner*), an den ein Beamer angeschlossen ist, durchführt. Auf dem Teachlet-Rechner sind eine Präsentationssoftware und mindestens eine Entwicklungsumgebung installiert. Der Moderator erklärt anhand der Entwicklungsumgebung das Ausgangssystem, eventuell unterstützt durch Folien. Er erklärt außerdem das Lernziel und erläutert die Aufgabenstellung. Anschließend fungiert er überwiegend passiv und setzt die Anweisungen um, die ihm von den Teilnehmern der Teachlet-Einheit zur Lösung der Aufgabe gegeben werden. Er kann jederzeit moderierend eingreifen, wenn die Vorschläge zu sehr vom gewünschten Lernziel wegführen; er sollte jedoch auf keinen Fall die Lösung „herunterprogrammieren“. Üblicherweise hat ein Moderator eine „Choreographie“ im Kopf, nach der das Teachlet durchgeführt wird. Er sollte auch immer eine fertige Lösung in der Hinterhand haben, die er im Notfall präsentieren kann.

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

2.3 Eine typische Choreographie

Üblicherweise wird das Ausgangssystem zu Anfang in eine kleine Story eingebettet, insbesondere dann, wenn aus didaktischen Gründen starke Vereinfachungen gegenüber einem realistischen System vorgenommen wurden. Dann startet der Moderator die Anwendung, führt selbstständig eine möglichst kurze Demo durch und ermuntert dann die Teilnehmer dazu, ihn per Zuruf zu weiterer oder wiederholender Erkundung der Funktionalität aufzufordern. Auf diese Weise lernen die Teilnehmer bereits die Funktionalität interaktiv kennen.

Erst wenn der Leistungsumfang klar geworden ist, sollte die Architektur untersucht werden. Insbesondere hier tritt der Moderator in den Hintergrund („Für welche Klasse soll ich zuerst den Quelltext zeigen?“) und folgt den Anweisungen der Teilnehmer („Starte bitte noch einmal die Anwendung, das möchte ich ausgeführt sehen“ etc.).

Nachdem das Ausgangssystem aus Außen- und Innensicht allen Teilnehmern klar geworden ist, stellt der Moderator die (erste) Aufgabe. Idealerweise ist diese auf mindestens einer eigenen Folie formuliert. Meist besteht die Aufgabe in einer Erweiterung der Funktionalität („ein *Undo* soll möglich sein“), aber das muss nicht der Fall sein. Gerade Entwurfsmuster werden häufig dann eingesetzt, wenn die innere Architektur nicht ausreichend wartbar ist. Eine Aufgabe kann deshalb auch lauten, die Anzahl der Klassen zu minimieren oder die Austauschbarkeit eines Quelltextabschnitts zukünftig einfacher zu gestalten.

Für die sehr interaktive Implementierungsphase sollten mindestens 30 Minuten eingeplant werden. Der Moderator muss dabei immer im Auge behalten, dass genügend Zeit für eine abschließende Zusammenfassung bleibt. In dieser Zusammenfassung sollte, unterstützt durch geeignete Folien, das gerade Getane noch einmal ausführlich reflektiert werden. Außerdem sollten hier alternative Lösungen und weitere Einsatzmöglichkeiten des zu lernenden Konzepts dargestellt werden. Im Abschluss wird eine Teachlet-Einheit somit meist wieder zu einer klassischen Präsentation.

Idealerweise sollten allen Teilnehmern nach einer Teachlet-Einheit die Quelltexte der gemeinsam erarbeiteten Lösung zur Verfügung gestellt werden, damit sie bei Bedarf die Möglichkeit haben, sich noch einmal individuell mit dem Gelernten auseinander zu setzen.

3 Ein Beispiel

Um die theoretische Darstellung des letzten Abschnitt zu verdeutlichen, wird in diesem Abschnitt ein Beispiel beschrieben. Es handelt sich um ein Teachlet für die Vermittlung des *Befehlsmusters*. Es wurde in der in Abschnitt 5 beschriebenen Musterwerkstatt entwickelt und bereits einmal außerhalb dieser Veranstaltung eingesetzt.

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

3.1 Die Exposition

Die Teachlet-Einheit beginnt mit der Vorführung des Ausgangssystems. Abbildung 1 zeigt seine grafische Benutzungsoberfläche. Es handelt sich um ein sehr einfaches System zum Üben der vier Grundrechenarten, das in dieser Form beispielsweise für Kinder im Grundschulalter geeignet sein könnte.

Die Oberfläche besteht aus einem Fenster, das in der oberen Zeile fünf Knöpfe enthält: die Symbole der vier Grundrechenarten sowie das Gleichheitszeichen. In der zweiten Zeile folgt die Anzeige eines so genannten Ausgangswertes (zu Beginn 1) sowie eines zufällig gewählten Operanden. Es folgen zwei anfangs funktionslose Knöpfe: Der erste ist mit einem Undo-Pfeil unterlegt, während der zweite die Anzeige eines Protokolls ermöglichen soll, das in der noch leeren Listbox rechts unten darzustellen ist.

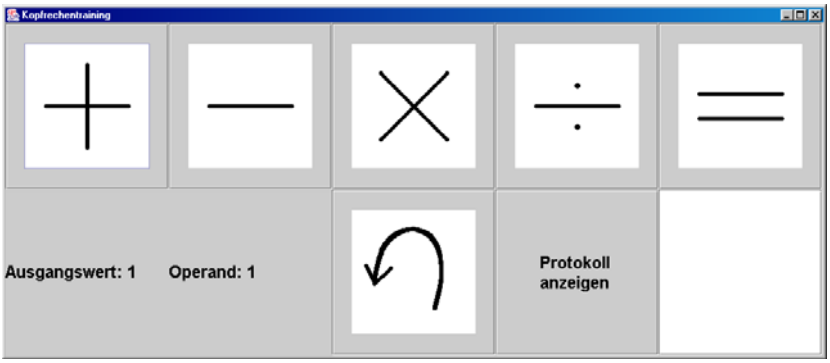

Abb. 1.: Die grafische Oberfläche des Ausgangssystems

Das eigentliche Kopfrechentraining wird nun wie folgt durchgeführt: Der Anwender wählt per Knopfdruck einen der vier Operatoren aus, sodass dieser im Hintergrund auf die „Ausgangswert“ genannte Zahl als ersten Operanden und den mit „Operand“ bezeichneten Wert als zweiten Operanden angewandt wird. Dieses Ergebnis stellt im nächsten Rechenschritt den ersten Operanden dar, während unter „Operand“ ein neuer zufällig gewählter Wert von 1 bis 9 erscheint, welcher wiederum den zweiten Operanden der folgenden Operation repräsentiert. Der ursprüngliche Ausgangswert wird mittlerweile ausgegraut, um zu verdeutlichen, dass er nicht mehr direkt in die folgenden Operationen involviert ist. Nach Klick auf das Gleichheitszeichen wird das aktuelle Resultat schließlich angezeigt, und zwar wieder schwarz unter „Ausgangswert“. Bei der nächsten Folge von Rechenoperationen stellt er folglich auch den neuen Ausgangswert dar.

Das Programm wird gestartet und der Moderator führt die Funktionalität einmal eigenständig vor. Anschließend ermuntert er die Teilnehmer, ihn bei der Benutzung anzuleiten, um mit ausreichenden Wiederholungen allen Teilnehmern die Funktionalität klar zu machen. Dieser einleitende Abschnitt sollte ca. 5 bis höchstens 10 Minuten dauern. Auf einer Folie wird die Funktionalität noch einmal zusammenfassend dargestellt.

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

Im nächsten Schritt erfolgt die Code-Inspektion. Das System besteht aus drei Klassen, die ausführlich und interaktiv betrachtet werden. Alle Teilnehmer sollten die interne Struktur der Anwendung verstanden haben. Auch dieser Abschnitt sollte nicht länger als 10 Minuten dauern.

3.2 Aufgabe, Lösungsvorschlag, Implementierung

Nachdem alle die Ist-Situation kennen, kann die Aufgabe gestellt werden. Es soll das noch fehlende Protokoll implementiert werden, das die geklickten Operatoren, jeweils mit ihrem Operanden, auf Knopfdruck (*Protokoll anzeigen*) darstellt. Außerdem sollen geklickte Rechenoperationen auch wieder zurückgenommen werden können. Diese Aufgabenstellung wird auf einer eigenen Folie dargestellt.

Nachdem die Aufgabe vermittelt wurde, schlägt der Moderator den Einsatz des Befehlsmusters zur Lösung vor. Die Aufrufe an Objekte der Klasse `Zwischenergebnis` mit ihren Berechnungsoperationen sollen quasi „objektifiziert“ werden. Diese Befehlsobjekte können dann gespeichert werden, in lesbarer Form angezeigt und auch gelöscht werden. Der Moderator zeigt eine Folie mit dem allgemeinen Klassendiagramm des Befehlsmusters und bildet dieses dann gemeinsam mit den Teilnehmern auf einen konkreten Entwurf an der Tafel ab. Für diesen Abschnitt sollen 10 bis 15 Minuten eingeplant werden.

Wenn sich alle Teilnehmer auf die Klassennamen im Entwurf geeinigt haben, wird der Entwurf gemeinsam implementiert. Je nach Programmierkultur kann inkrementell iterativ oder in einem großen Entwicklungsschritt vorgegangen werden, und je nach vorhandener Zeit kann auch noch ein Makrobefehl implementiert werden. Für diesen Abschnitt sollten mindestens 30 Minuten eingeplant werden.

3.3 Zusammenfassung, Abschluss

Der Entwurf und die Implementierung werden auf vorbereiteten Folien reflektiert. Auf einer Folie sind die Teilnehmer des generischen Musters den Teilnehmern der konkreten Realisierung gegenübergestellt (bei der die Namen durchaus von den in der Gruppe gewählten abweichen können). Eine weitere Folie zeigt das spezifische Klassendiagramm der Lösung, bei der auch sehr wahrscheinlich die Namen abweichen. Schließlich werden die Vorteile des Befehlsmusters und weitere Anwendungsmöglichkeiten auf Folien zusammengefasst. Dies alles sollte nicht länger als 10 Minuten dauern.

4 Bisherige Erfahrungen mit Teachlets

Bei den bisherigen Erfahrungen mit Teachlets konnten unterschiedliche Muster bei der Choreographie beobachtet werden.

Einige Teachlets wurden direkt mit dem Ausgangssystem begonnen, ohne einführende Folien zum Lernziel (keine „trockene“ Darstellung eines Entwurfsmusters zu Anfang). Die Aufgabe wurde bearbeitet und anschließend wurde „enthüllt“, dass da

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

gerade etwas benutzt wurde, das üblicherweise als XY (beispielsweise als Adapter) bezeichnet wird. Dies war meist dann sinnvoll, wenn das Entwurfsmuster selbst nicht sehr kompliziert ist und primär der Terminologiebildung dient.

Andere Teachlets wurden mit einer Beschreibung des Lernziels eingeleitet, weil das Entwurfsmuster zu anspruchsvoll ist, um von den Teilnehmern innerhalb der Lerneinheit selbstständig abgeleitet zu werden. Dies war bei einem Teachlet zum Interpretermuster so und würde vermutlich auch etwa beim Besuchermuster so sein. Hier wird die grundlegende Struktur des Musters zuerst vorgestellt und anschließend das Muster auf eine konkrete Problemstellung angewendet.

Abweichend von der Darstellung im letzten Abschnitt kann es auch sinnvoll sein, die Aufgabe zu stellen, bevor die Architektur des Ausgangssystems untersucht wurde. Wenn die Aufgabe sich ausschließlich auf die Funktionalität bezieht, kann es für die Teilnehmer sogar sinnvoller sein, sich problemorientiert in den Quelltext einzulesen.

4.1 Hinweise für Moderatoren

Die Moderation eines Teachlets ist anspruchsvoll; anspruchsvoller als beispielsweise ein reiner Folienvortrag. Grundsätzlich gelten die üblichen Hinweise zu Folienvorträgen auch für Teachlets, insbesondere für den Folienanteil.

Ein Teachlet wird häufig im Sitzen moderiert. Für den Folienanteil ist das möglicherweise ungewohnt, es sollte dem Moderator vorher bewusst sein. Der Moderator sollte auch den Teachlet-Rechner kennen, um Tastenkombinationen (wie Strg-v) flüssig benutzen zu können.

Aus den bisherigen Erfahrungen mit Teachlets konnten die folgenden Hinweise für Teachlet-Moderatoren destilliert werden:

- *Mache die Aufgabenstellung so deutlich wie möglich.* Wenn die Aufgabe nicht gut von allen Teilnehmern verstanden ist, kommt ein Teachlet oft ins Stocken. Idealerweise ist die Aufgabe auf eigenen Folien formuliert. Es sollte explizit nachgefragt werden, ob alle Teilnehmer die Aufgabenstellung verstanden haben; sich nicht äußernde Teilnehmer sollten explizit zum Nachfragen ermuntert werden, wenn noch nicht alles klar ist.
- *Halte die Stille aus.* Wenn bei einem Teachlet die Teilnehmer ruhig sind, dann oft deshalb, weil sie sich in den Quelltext einlesen bzw. eindenken. Das braucht Zeit, die auch gelassen werden sollte.
- *Behalte den Überblick und die Kontrolle.* Häufig gibt es ausführliche Diskussionen unter den Teilnehmern. Diese sind erwünscht, sollten aber nicht zu sehr ausufern. Notfalls müssen schlechte Beiträge knapp behandelt und langwierige Diskussionen beendet werden. Die Choreographie sollte nicht aus den Augen verloren werden. Als guter Tipp hat sich erwiesen: Die Rückkehr zum Ablauf sollte nicht mit der Vorwegnahme des nächsten Schrittes erzwungen werden, sondern mit einer Zusammenfassung des bisher Geschehenen und der Frage, wie es nun weitergehen soll.

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

- *Denke laut*. Es sollten alle Dinge erläutert werden, die in der Entwicklungsumgebung passieren. Alle eingesetzten Werkzeuge sollten bei ihrer Benutzung kommentiert werden („So, jetzt übersetze ich das mal“, „An dieser Stelle setze ich ein Refactoring von Eclipse ein“ etc.). Insbesondere eher ungewöhnliche oder seltene Tätigkeiten müssen erläutert werden.
- *Beherrsche den Lösungsraum*. Für eine spezifische Aufgabenstellung gibt es oft mehrere Lösungen. Der Moderator sollte gut auf verschiedene Lösungsvorschläge vorbereitet sein. Er sollte sich gute Argumente gegen Lösungen zurechtlegen, die vom Lernziel ablenken würden, aber andererseits auch offen sein für gute Vorschläge, die ihm selbst vorher nicht eingefallen sind.
- *Setze gezielt Wiederholungen ein*. Wiederholungen ermöglichen Teilnehmern, die kurz abgeschaltet haben, wieder „auf die Spur“ zu kommen. Zusammenfassungen sollten immer wieder eingestreut werden, in denen der Moderator das bisherige Geschehen knapp wiederholt. Sie können oft mit dem Satz eingeleitet werden: „Was haben wir bisher gemacht?“
- *Benutze Klassendiagramme zur Visualisierung*. Wenn ein Standardmuster aus [GHJV95] vorgestellt wird, dann sollte auf den Folien zumindest das generische Klassendiagramm des Musters aufgeführt sein. Für das konkrete System ist außerdem oft ein spezifisches Klassendiagramm wünschenswert. Je nach Teachlet ist entweder ein Diagramm des Ausgangssystems oder ein Diagramm des fertigen Systems sinnvoll.
- *Beherrsche die notwendigen Grundlagen deines Teachlets*. Wenn sich etwa für die Umsetzung des Entwurfsmusters *Iterator* die inneren Klassen von Java anbieten, dann sollte der Moderator erstens gut über innere Klassen Bescheid wissen und zweitens Folien in der Hinterhand haben, mit deren Hilfe er die notwendigen Grundlagen bei Bedarf erläutern kann.

5 Eine Teachlet-Werkstatt: Lernen durch Lehren

„I hear, and I forget.
I see, and I remember.
I do, and I understand.“ (nach Konfuzius)

Dieses Sprichwort spiegelt die insbesondere in der Informatik gültige Erkenntnis wider, dass ein dauerhaftes Verständnis vor allem durch eigenes Handeln im Problembereich gefördert wird (siehe etwa [Ell98]). Als Motivation für die eingangs erwähnte Veranstaltung zu fortgeschrittenen Konzepten objektorientierter Programmierung wurde das Sprichwort durch folgenden Satz ergänzt:

„I teach, and I really understand.“

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

Die Grundidee der Veranstaltung beruht somit auf der Annahme, dass eine Person, die anderen Personen ein Konzept in einer Lehrveranstaltung vermitteln muss, noch stärker zur Reflexion gezwungen ist als eine ausschließlich handelnde Person.

Die Definition des Begriffs *Teachlet-Werkstatt* in Abgrenzung zum Teachlet-Begriff lautet:

„Eine Teachlet-Werkstatt ist eine seminarartige Semesterveranstaltung, in der die Teilnehmer neue Teachlets ausarbeiten. Jeder Teilnehmer muss sich dazu mit mindestens einem Lernziel befassen und für dieses Lernziel ein Teachlet entwerfen. Die neuen Teachlets werden an den Teilnehmern der Werkstatt selbst ausprobiert und anschließend von den Teilnehmern analysiert und bewertet.“

Eine Teachlet-Werkstatt ist also deutlich gegenüber einer fiktiven Lehrveranstaltung beispielsweise zu Entwurfsmustern abgegrenzt, in der der Dozent ausschließlich mit bewährten Teachlets arbeitet. Der bereits hohe interaktive Anteil einer Teachlet-Veranstaltung wird somit in einer Teachlet-Werkstatt weiter gesteigert.

5.1 Erstmalige Durchführung

Die in Abschnitt 2 dargestellte Definition für Teachlets wurde den Teilnehmern in der Einführungsveranstaltung vorgestellt. An den folgenden beiden Terminen führte der Veranstalter zwei Teachlets selbst durch, um das Konzept zu verdeutlichen. Zu den restlichen Terminen führten die Teilnehmer nur noch Teachlets durch, die sie selbst ausgearbeitet hatten. Bei der Vorbereitung wurden sie ähnlich intensiv betreut, wie es in klassischen Seminaren oft der Fall ist: Mindestens zwei Wochen vor dem Termin musste dem Veranstalter ein Grobkonzept vorgestellt werden, spätestens eine Woche vorher mussten alle statischen Teile des Teachlets präsentiert werden.

Alle Moderationen wurden allein durchgeführt. Nach jeder Teachlet-Einheit gab es intensives Feedback auf den unterschiedlichen Ebenen: zum Lernziel selbst, zum Teachlet sowie zur Moderation. Sowohl die Teilnehmer selbst als auch der Veranstalter haben jeweils Feedback gegeben.

5.2 Beobachtungen

Der Versuch, eine Unterrichtseinheit zu einem Entwurfsmuster als Teachlet zu gestalten, führt tatsächlich zu einer vertieften Beschäftigung mit dem Lernziel. Alle studentischen Teachlet-Moderatoren haben ihren Stoff gut beherrscht.

Nicht jede Person ist unmittelbar zum Teachlet-Moderator geeignet. Voraussetzung ist unter anderem ein flüssiger Vortragsstil bereits bei Folienvorträgen. Diese Voraussetzung wurde nicht von allen Teilnehmern gleich gut erfüllt. Darüber hinaus werden aber noch weitere Soft Skills benötigt: die ständige Reflexion darüber, welcher Einwand wie nützlich für das Vorankommen ist, sowie wiederholtes Zusammenfassen des bisher Geschehenen – denn der Moderator muss jederzeit den Ablauf im Blick haben. Die Fähigkeit, starke Präsenz in einer Diskussion zu zeigen – denn der Moderator muss jederzeit den Ablauf unter Kontrolle haben. Dies sind Fähigkeiten, die jedem Software-

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

techniker zugute kommen. Bei der Moderation eines Teachlets wird schnell deutlich, in welchen dieser Bereiche noch Defizite existieren und an welchen Stellen somit noch gearbeitet werden sollte.

Auf der Meta-Ebene wurden etliche Diskussionen über den Programmierstil, aber auch über die jeweils verwendete Entwicklungsumgebung (im Folgenden kurz IDE) geführt. Zum Einsatz kamen *Eclipse* [Eclipse], *IntelliJ IDEA* [Jetbrains] und *BlueJ* [BlueJ]. Insbesondere über die professionell ausgerichteten Umgebungen Eclipse und IDEA gab es am Rande immer wieder kurze Diskussionen. Nach Eindruck des Veranstalters waren diese Diskussionen ausgesprochen fruchtbar, da sie unterstrichen, wie wichtig das Werkzeug IDE für den Softwaretechniker ist. Es entstand eine Atmosphäre des Austauschs über Unterstützungsmöglichkeiten durch die IDE (insbesondere für Refactorings), die in einer Veranstaltung über objektorientierten Entwurf durchaus angemessen war.

6 Bewertung und Ausblick

6.1 Das Teachlet-Konzept

Bewertung

Das Teachlet-Konzept selbst hat sich für die Vermittlung von Entwurfsmustern in Gruppen von bis zu 16 Teilnehmern bewährt. Das Feedback der Teilnehmer hat deutlich gemacht, dass ein *problembasierter Lehransatz* ([Ell98]) gerade zum Aneignen von Entwurfsmustern ausgesprochen motivierend ist. Die Vorteile einer engen Verzahnung von theoretischem Vortrag und praktischer Arbeit in der Programmierausbildung sind kürzlich auch von Kölling und Barnes in [KöB04] beschrieben worden. Das Teachlet-Konzept löst die traditionelle Trennung von Vorlesung und Übung noch weiter auf und stellt den Umgang mit Software-Artefakten in den Mittelpunkt einer hochinteraktiven Lehr- bzw. Lerneinheit. Teachlets ermöglichen somit die Gestaltung sehr effektiver Veranstaltungen.

Kölling und Barnes sprechen in [KöB04] außerdem einen interessanten Aspekt an, der auch bei Teachlets eine starke Bedeutung hat: Der Teachlet-Moderator kann als Rollenmodell dienen, das gute Praktiken der Softwareentwicklung vorlebt.

Als Nachteil von Teachlets ist zu bemerken, dass sie ausschließlich in echten Lehrsituationen evaluiert werden können. Auf diese Weise „reifen“ sie nur sehr langsam.

Ausblick

Zukünftig gilt es zu untersuchen, inwieweit die zu vermittelnden Inhalte und die Teilnehmerzahl eines Teachlets variiert werden können:

- Es wurde bisher nur wenig Erfahrung bei der Vermittlung von Programmiersprachkonzepten gesammelt, obwohl dies ursprünglich für die Teachlet-

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

Werkstatt vorgesehen war. Aufgrund der Wahlmöglichkeiten der Teilnehmer gab es keine Teachlets zu beispielsweise multipler (Implementierungs-) Vererbung, Generizität oder Multi-Methoden. Hier müssen weitere Erfahrungen gesammelt werden. Auch das Vermitteln von Algorithmen könnte durch den Einsatz von Teachlets interessanter gestaltet werden, insbesondere weil die Einbettung eines Algorithmus in einen Problemkontext vermutlich stark motivierend für die Studierenden ist.

- Es sollte untersucht werden, inwieweit der Ansatz skaliert, d.h. mit welcher Anzahl von Teilnehmern das Konzept maximal umsetzbar ist. Wenn die Gruppe sehr groß wird, kann sich nicht mehr jeder Teilnehmer einbringen. Trotzdem könnte der Einsatz von Teachlets auch bei größeren Gruppen fruchtbar sein, weil die passiven Teilnehmer an der interaktiven Problemlösung, die von den aktiveren Teilnehmern vorangetrieben wird, vermutlich besser teilhaben können als an einem reinen Frontalvortrag. Hier müssen noch weitere Versuche unternommen werden.

Teachlets, die sich als zielführend erwiesen haben und einen ausreichenden Reifegrad erreicht haben, sollten anderen Lehrenden zur Verfügung gestellt werden. Denkbar ist eine koordinierte Sammlung von Teachlets zu unterschiedlichsten Lernzielen, die über einen dedizierten Webauftritt zugänglich gemacht wird, ebenso wie eine Einbindung in beispielsweise MuSofT [Alf03], einem deutschen Portal für multimediales Lehrmaterial in Softwaretechnik-Veranstaltungen.

Da Teachlets bisher ausschließlich von der Person durchgeführt wurden, die sie ausgearbeitet hat, muss außerdem untersucht werden, wie Teachlets geeignet dokumentiert werden können. Ziel sollte sein, dass eine ausreichend kompetente Person ein ihr unbekanntes Teachlet mit minimalem Vorbereitungsaufwand durchführen kann.

6.2 Die Teachlet-Werkstatt

Bewertung

In einer eigenen Veranstaltung wurde das Erstellen von Teachlets in den Mittelpunkt gestellt. Die relativ starke Formalisierung des Teachlet-Konzepts hat dabei den Studierenden die Arbeit erleichtert.

Die Erstellung eines guten Teachlets stellt eine hohe Motivation für Studierende dar. Sie erfordert Kreativität für die grundlegende Idee und handwerkliches Können für die Entwicklung des Ausgangssystems. Die Durchführung des Teachlets erfordert außerdem ein souveränes Umgehen mit einer IDE samt Folien und Beamer vor einer Gruppe, dabei werden diverse Soft Skills trainiert. Und schließlich besteht die Aussicht, dass das Ergebnis einem größeren Kreis zugänglich gemacht werden könnte. Entsprechend hoch war die Motivation der Teilnehmer an der Teachlet-Werkstatt.

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

Ausblick

Eine Wiederholung der Veranstaltung ist für das Sommersemester 2005 geplant. Einige der Teachlets der ersten Veranstaltung werden dabei in überarbeiteter Form erneut zum Einsatz kommen.

Das Bedienen eines Rechners mit Folien und IDE erfordert viel Aufmerksamkeit, die Moderation der fachlichen Diskussion in einer größeren Gruppe ebenfalls. Es wäre deshalb denkbar, die Moderation im Paar durchführen zu lassen. Für den breiten Einsatz von Teachlets in Lehrveranstaltungen ist dies vermutlich nicht tragfähig, für eine Lehr-Werkstatt hingegen schon. In einer ersten Runde könnten die Studierenden neue Teachlets in Paaren ausarbeiten und moderieren und erst bei ihrem zweiten Teachlet allein moderieren.

7 Danksagungen

Ich danke allen Teilnehmern der ersten Teachlet-Werkstatt für ihre engagierte Mitarbeit, die das Konzept weiter bestätigt hat. Meinen Kolleginnen und Kollegen im Arbeitsbereich Softwaretechnik an der Universität Hamburg danke ich für einige fruchtbare Diskussionen über Teachlets und gute Anregungen für Abwandlungen. Den Anstoß zur Teachlet-Idee verdanke ich Heinz Züllighoven, der mir für einen unerwartet ungefüllten Seminartermin vor zwei Jahren vorschlug, eine kleine, interaktive Musterwerkstatt durchzuführen.

Literatur

[Alf03] Alfert, K., et al.: "MuSofT: Multimedia in der Softwaretechnik", Proc. Software Engineering im Unterricht der Hochschulen (SEUH), Berlin, S. 70-80, 2003.

[BlueJ] BlueJ – The Interactive Java Environment, *http://www.bluej.org*.

[CHV00] Chambers, C., Harrison, B., Vlissides, J.: "A Debate on Language and Tool Support for Design Patterns", Proc. 27th ACM SIGPLAN-SIGACT symposium on principles of programming languages, Boston, MA, S. 277-289, 2000.

[Eclipse] Eclipse – An Open Platform for Tool Integration, *http://www.eclipse.org*.

[Ell98] Ellis, A., et al.: "Resources, Tools, and Techniques for Problem Based Learning in Computing", Proc. ITiCSE '98, Dublin, Ireland, S. 46-50, 1998.

[GHJV95] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading, MA, 1995.

[Jetbrains] IntelliJ IDEA – An Intelligent Java IDE, *http://www.jetbrains.com/idea*.

[KöB04] Kölling, M., Barnes, D. J.: "Enhancing Apprentice-Based Learning of Java", Proc. SIGCSE 36, Norfolk, Virginia, S. 286-290, 2004.