Das Teachlet-Konzept: Möglichkeiten und Grenzen einer Lehrform für Software-Entwurfsdiskussionen

Julian Fietkau

6. Dezember 2010

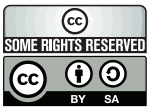
Bachelorarbeit

Zur Erlangung des akademischen Grades
Bachelor of Science. (B.Sc.)

Erstbetreuer: Dr. Axel Schmolitzky
Zweitbetreuer: Prof. Dr. Horst Oberquelle

Fachbereich Informatik
Universität Hamburg

Inhaltsverzeichnis



Zusammenfassung

Teachlets als innovative Lehrmethode befinden sich seit 2004 im Praxiseinsatz, Literatur ist zum Thema allerdings kaum vorhanden und nicht auf dem aktuellen Stand. In dieser Arbeit wird die Teachlet-Praxis der letzten Jahre resümiert, die bisherige Definition des Begriffs auf ihre Zukunftsfähigkeit untersucht und erweitert, aktuelle Entwicklungen dargestellt und konzeptuelle Grenzen sowie weitere Möglichkeiten von Teachlets als Lehrform für Inhalte der Softwaretechnik diskutiert.

1. Einführung

Teachlets sind ein am Fachbereich Informatik der Universität Hamburg entwickeltes Lehrkonzept für Inhalte der praktischen Informatik, das im Spannungsfeld zwischen theoretischem Vortrag und praktischer Programmierübung existiert und dabei versucht, die Vorteile beider Herangehensweisen zu aggregieren. Bei einem Teachlet gibt es vortragsartige Abschnitte, aber auch offene Entwurfsdiskussionen im Plenum, die vom Moderator geleitet live auf dem Präsentationsrechner in Code umgesetzt werden.

Durch die spezielle Methodik von Teachlets werden die Teilnehmer eines Teachlets stärker dabei gefördert, das Vorgestellte sofort zu reflektieren und anzuwenden, als bei einem traditionellen Vortrag[Sch07].

Das Teachlet-Konzept wurde 2004 erstmalig erprobt und 2005 veröffentlicht[Sch05]. Seitdem erfreut es sich stetig wachsender Popularität und wird von verschiedenen Moderatoren in der Praxis eingesetzt. Am Fachbereich Informatik der Universität Hamburg wird seit 2004 jedes Jahr eine *Teachlet-Werkstatt* durchgeführt, in der neue Teachlets für seminarähnliche Umfelder entworfen und erprobt werden. Außerhalb dieser Veranstaltung werden Teachlets vereinzelt in anderen universitären und außeruniversitären Kontexten eingesetzt.

1.1. Ziele dieser Arbeit

Für das Teachlet-Konzept existiert bisher nur wenig ausgereifte Dokumentation. Seitdem Axel Schmolitzky den Grundstein[Sch05] gelegt hat, hat es abgesehen von Experimenten mit Teachlets außerhalb der Softwaretechnik[BB07] zwar viele Teachlets gegeben, aber kaum belastbare Dokumentation zum Thema an sich.

In dieser Arbeit sollen die stattgefundenen Jahre des praktischen Einsatzes von Teachlets durch die Auswertung von Erfahrungsberichten evaluiert werden. Weiterhin wird überprüft, inwieweit die existierende Definition des Konzepts heutigen Anforderungen und Erfahrungen gerecht wird, um die gesammelten Erkenntnisse in eine überarbeitete Definition einfließen lassen. Mit dieser Definition wird dann weitergearbeitet und es werden weitere Impulse für das geliefert, was das Konzept in Zukunft weiter leisten kann und wo seine Grenzen liegen.

Obwohl der Einsatz von Teachlets sich evtl. auch in vielen anderen Bereichen lohnen könnte (siehe Abschnitt 6.1), bleibt diese Arbeit ausdrücklich auf den Kontext der softwaretechnischen Lehre beschränkt, aus dem die Teachlets ursprünglich stammen.

2. Die Teachlet-Idee

Das Grundziel von Teachlets ist das erfolgreiche Verweben von Theorie und Praxis in der Softwaretechnik-Lehre.

Bei einem rein theoretischen Vortrag gibt es einen oder mehrere *Vortragende*, die vor einem *Publikum* aktiv sind und Wissen präsentieren. Das Publikum ist passiv und nimmt lediglich Informationen auf. Im Gegensatz dazu sind bei einer praktischen Übung die Übungs*teilnehmer* aktiv, während einer oder mehrere *Betreuer* parallel oder zeitversetzt ihren Fortschritt überprüfen.

Bei einem Vortrag hat der Vortragende eine sehr genaue Kontrolle darüber, was und wie viel vermittelt wird. Die Zuhörenden bekommen alle die gleichen Informationen und haben die gleichen Lernchancen. Allerdings sind lange, trockene Vorträge aus lernpsychologischer Sicht problematisch, da das Wissen nur selten richtig aufgenommen und gefestigt wird, wenn es nicht in einen praktischen Kontext übertragen wird.

Durch Übungen können theoretische Erkenntnisse erprobt und verfestigt werden. Dadurch, dass die Teilnehmer aktiv handeln statt nur zu konsumieren, können sie nachhaltiger lernen. Ein typisches Hindernis bei der Durchführung von Übungen ist die unterschiedliche Geschwindigkeit der Teilnehmer, die es erschwert, einen ungefähr einheitlichen Stand zu wahren, ohne dass sich einzelne Teilnehmer über- oder unterfordert fühlen.

Um von den positiven Aspekten beider Lehrformen zu profitieren und die negativen so weit wie möglich einzuschränken, wurde das Teachlet-Konzept entwickelt. Bei einem Teachlet gibt es einen oder mehrere Moderatoren, deren Rolle zwischen aktiven und passiven Abschnitten wechselt. Die Teilnehmer bekommen teilweise Informationen präsentiert, wenden diese dann jedoch sofort praktisch an. Dies geschieht nicht bei jedem einzelnen Teilnehmer für sich, sondern kollaborativ mit der gesamten Gruppe auf Basis einer gemeinsamen Sicht auf die Software (in der Regel ist das ein Präsentationsrechner mit Beamer).

Die aktive Beteiligung der Teilnehmer, also ein sehr hoher Grad der Interaktivität, ist dabei eines der kennzeichnenden Elemente eines Teachlets und unterscheidet es von einem Vortrag. Die Teilnehmer sollen dabei die Lösung für das gegebene Problem nicht vorgesetzt bekommen, sondern sie tatsächlich selbst erarbeiten.

Eine ausführliche Beschreibung und Begründung der Grundidee findet sich im ursprünglichen Teachlet-Paper[Sch05]. Eine Einordnung des Konzepts für Lehrende im Bereich der Softwaretechnik existiert ebenfalls bereits[Sch07].

2.1. Ein ausführliches Beispiel

Ein Lehrkonzept erschließt sich nur schwierig durch abstrakte Beschreibungen und theoretische Erläuterungen. Seine Praxistauglichkeit zeigt sich letztlich erst ebendort, in der Praxis. Die beste Vorgehensweise zum Kennenlernen von Teachlets wäre in diesem Sinne der Besuch einer Teachlet-Einheit. Leider ist dies nicht immer auf Anhieb möglich.

Als Ersatz für das Miterleben soll hier eine vergangene Teachlet-Einheit detailliert wiedergegeben werden in der Hoffnung, dass es dem tieferen Verständnis dient.

Dieses ausführliche Beispiel wird hierbei bewusst bereits vor der Begriffsklärung gegeben, damit eine anschauliche Vorstellung von Teachlets existiert, bevor alle Einzelaspekte erläutert werden. Falls beim Lesen des Beispiels Unklarheiten auftreten, können einzelne Begriffe zwischendurch im Abschnitt 2.2 nachgeschlagen werden.

Die im Folgenden vorgestellte Teachlet-Einheit fand am 27. April 2010 in der Teachlet-Werkstatt an der Universität Hamburg statt, basierend auf einem Teachlet, das 2009 in der Teachlet-Werkstatt in Zusammenarbeit mit einer Kommilitonin vom Autor dieser Arbeit erstellt wurde. Das Teachlet befasst sich mit dem Entwurfsmuster *Zustand* und trägt den Titel „Sessel-o-matic“. Die Einheit wurde mit mehreren Videokameras aufgezeichnet, was es ermöglicht, den Ablauf hier sehr genau wiederzugeben.

Begrüßung und Szenario

Der Moderator begrüßt die Teilnehmer und stellt sich kurz vor. Danach beschreibt er den Teilnehmern die Situation des Teachlets: Die Teilnehmer sollen einem Hersteller von Massagesesseln bei der Programmierung seiner Sessel helfen, da das bisherige Programmierteam es nicht geschafft hat, die Spezifikation umzusetzen. Gezeigt wird die in Java umgesetzte Fernbedienung für den Sessel (Abbildung 1). Der Moderator erläutert die Funktionen des Sessels: Es gibt eine Fußstütze, die hoch- oder runtergestellt sein kann; eine Lehne, die aufrecht- oder zurückgestellt sein kann sowie eine Massagefunktion, die aus- oder eingeschaltet sein kann. Die Fernbedienung hat Knöpfe für diese sechs Funktionen und ein Display für den aktuellen Zustand des Sessels.

Erkunden der Funktionalität des Ausgangssystems

Die Teilnehmer erhalten die Gelegenheit, die bisherige Funktionalität der Software zu erkunden. Dazu startet der Moderator das System auf dem Präsentationsrechner und fragt die Teilnehmer, welchen Button er klicken soll. Auf Zuruf führt er die Wünsche der Teilnehmer aus, so dass diese sich bereits nach kurzer Zeit erschließen, dass der Sessel nur bei zurückgestellter Lehne seine Massagefunktion anschalten lässt. Die Fußstütze lässt sich überhaupt nicht verstellen, Klicks auf diese beiden Buttons bleiben ohne Auswirkung.

Währenddessen kommentiert der Moderator das Verhalten des Programms um den Teilnehmern dabei zu helfen, keine falschen Schlussfolgerungen zu ziehen. Er resümiert kontinuierlich die Beobachtungen und konsolidiert mit den Teilnehmern die Kenntnis des aktuellen Zustandsraumes des Sessels (Abbildung 2).

Am Ende dieses Abschnitts haben die Teilnehmer verstanden, was das System bisher kann und wie es sich verhält.

Arbeitsauftrag und Erweiterung des Zustandsdiagramms

Der Moderator zeigt und erklärt die Aufgabenstellung: Die Teilnehmer sollen die Fußstützen verstellbar machen. Außerdem soll die Massage nur genau dann aktiviert werden können, wenn die Füße hoch- und die Lehne zurückgestellt sind.

Abbildung 1: Dieses Fenster sehen die Teilnehmer beim „Sessel-o-matic“-Teachlet, wenn das Ausgangssystem erstmalig gestartet wird. Die mittleren zwei Knöpfe sind noch an keine Funktionalität gebunden – diese soll im Laufe der Teachlet-Einheit nachgerüstet werden.

Abbildung 2: Dieses Zustandsdiagramm entspricht dem beobachteten Verhalten des Ausgangssystems, erfüllt jedoch nicht die Anforderungen an die Software. Bevor programmiert wird, müssen die Teilnehmer sich überlegen, wie sie den Zustandsraum erweitern und strukturieren möchten.

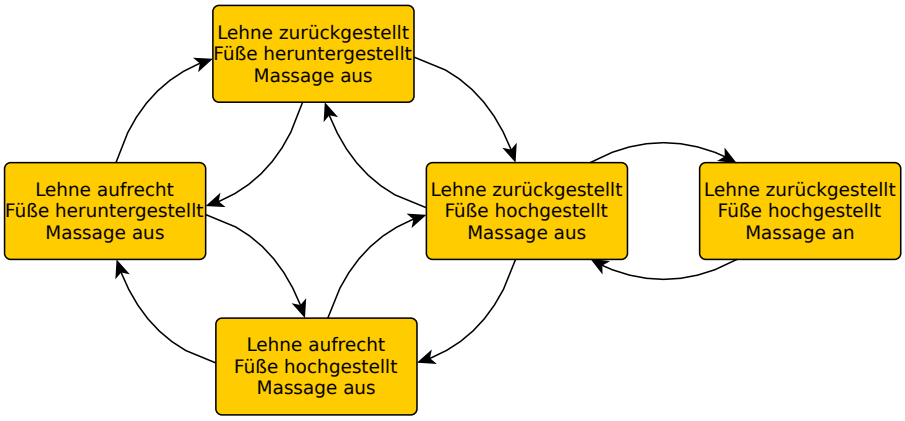


Abbildung 3: Dieses oder ein strukturell gleiches Zustandsdiagramm sollen die Teilnehmer aus den Anforderungen entwickeln. Es dient als Diskussions- und Arbeitsgrundlage für das weitere Vorgehen.

Im Dialog mit den Teilnehmern entwickelt der Moderator das Zustandsdiagramm, das erreicht werden soll (Abbildung 3). Bei diesem Diagramm handelt es sich um das vordergründige Ziel der Entwicklungsphase in dieser Teachlet-Einheit.

Nachdem alle Teilnehmer das Zustandsdiagramm verstanden haben, verteilt der Moderator ein Handout, auf dem ein Screenshot der Fernbedienung sowie die beiden Zustandsdiagramme abgedruckt sind. So haben die Teilnehmer diese drei Dinge jederzeit vor Augen.

Code-Inspektion

Der Moderator öffnet das Java-Projekt in der Eclipse-Entwicklungsumgebung. In vielen Teachlets passiert dies vor der Besprechung der Aufgabenstellung – dass der Code in diesem Teachlet erst vergleichsweise spät betrachtet wird, ist eine bewusste Entscheidung der Autoren des Teachlets gewesen und nicht unbedingt typisch.

Der Moderator fragt die Teilnehmer, welche Klasse sie als erstes sehen möchten. Auf Zuruf öffnet er nacheinander die vier Klassen des Projektes und erläutert nebenbei die grundsätzliche Funktionsweise und einige Stolperfallen des Codes. Die Interaktion von Fernbedienung und Sessel-Modell ist an das Beobachter-Muster angelehnt. Die Zustandsabfragen und -änderungen erfolgen mit verschachtelten if-Abfragen.

Die Teilnehmer verschaffen sich einen Überblick über den Code des Systems.

Klassendiagramm erstellen

An der Tafel wird von den Teilnehmern ein Klassendiagramm des Ausgangssystems entwickelt, wobei der Moderator die Kreide in der Hand hält und das schreibt und zeichnet, was die Teilnehmer ihm sagen. Die Verwendungs- und Vererbungsbeziehungen

werden eingetragen, dabei passiert zunächst ein Fehler, der jedoch von den Teilnehmern selbst rechtzeitig bemerkt und korrigiert wird.

Am Ende der Einheit steht an der Tafel ein korrektes Klassendiagramm des Ausgangssystems.

Entwurfsdiskussion

Der Moderator beginnt ein Gespräch zum Thema, wie die Anforderung umgesetzt werden kann. Die Teilnehmer diskutieren über Vor- und Nachteile verschiedener Ansätze hinsichtlich Kopplung und Wartbarkeit. Einige Teilnehmer sind grundlegend mit dem Zustandsmuster vertraut und erklären die Idee des Musters.

Diskutiert werden ein Ausbau des bestehenden Ansatzes, eine Umsetzung eines Entwurfs mit einer Adjazenzmatrix sowie mehrere Ideen, die in die Richtung des Zustandsmusters gehen. Die Entwurfsdiskussion wird unterbrochen, damit der Moderator das Entwurfsmuster vorstellen kann.

Vorstellung des Zustandsmusters

Innerhalb weniger Minuten stellt der Moderator das Entwurfsmuster „Zustand“ in seiner abstrakten Beschreibung vor und erwähnt Vor- und Nachteile. Die Teilnehmer stellen Fragen und diskutieren anhand des Ausgangssystems des Teachlets über die Eigenschaften des Musters.

Klassendiagramm erweitern

Die Teilnehmer einigen sich zunächst auf eine grobe Umsetzung anhand des Zustandsmusters, danach wird an der Tafel das bereits vorhandene Klassendiagramm um den Entwurf der Teilnehmer ergänzt, der in diesem Zuge konkretisiert wird: Die Teilnehmer legen Namen und Schnittstellen ihrer Klassen sowie deren Beziehungen untereinander fest.

Umsetzung

Nach der abgeschlossenen Planung öffnet der Moderator erneut die Entwicklungsumgebung. Im Ausgangssystem legt der Moderator in enger Rückkopplung mit den Teilnehmern die Zustandsklassen an. Er kennt die richtigen Abkürzungen in der Entwicklungsumgebung, um mit wenig Aufwand den nötigen Code für die Schnittstellen der Klassen zu erzeugen. Von den Teilnehmern kommen Programmlogik und Methodenrümpfe.

Die Implementierung gelingt nicht ganz vollständig innerhalb der Zeit, weshalb der Moderator kurz vor Ende ein fertiges Zielsystem öffnet, das nach einem nahezu identischen Entwurf umgesetzt wurde. Er betrachtet mit den Teilnehmern die Teile des Codes, die in ihrem System noch gefehlt hätten. Die Teilnehmer diskutieren noch über einige Feinheiten der Umsetzung. Die Funktionalität des Systems wird getestet und erfüllt die Anforderung.

Zusammenfassung

Der Moderator fasst erneut die Eigenschaften des Entwurfs nach dem Zustandsmuster zusammen, reflektiert mit den Teilnehmern die Umsetzung und beglückwünscht sie zur gelungenen Umsetzung. Die Teilnehmer haben im Laufe einer 90-minütigen Teachlet-Einheit das Zustandsmuster kennengelernt und in einem lauffähigen Java-System umgesetzt. Bevor die Teilnehmer gehen, gibt es noch eine kurze Diskussion über weitere Alternativen für die Umsetzung.

2.2. Begriffsklärung

Um den Diskurs zum Thema klar zu gestalten, sollen im Folgenden einige Begriffe genannt und definiert werden, die im Teachlet-Umfeld verwendet werden.

Ausgangssystem
: Eine lauffähige, aber auf eine ganz bestimmte Weise defizitäre Software. In der Regel ist diese speziell für ein →*Teachlet* entworfen, so dass durch die erfolgreiche praktische Anwendung des →*Lernziels* des Teachlets (z.B. ein bestimmtes objektorientiertes Entwurfsmuster) das genannte Defizit der Software im Laufe der →*Teachlet-Einheit* von den →*Teilnehmern* behoben werden kann.

Autor
: Zu einem →*Teachlet* gehören einer oder mehrere Autoren, die das Teachlet erstmalig konzipiert und dokumentiert haben. Normalerweise ist der Autor gleichzeitig der →*Moderator* bei der ersten Aufführung eines Teachlets. Ein Moderator kann allerdings auch Teachlets von anderen Autoren aufführen.

Choreographie
: Eine zeitliche Planung für Aufführungen eines bestimmten →*Teachlets*, die den Ablauf und zeitliche Grenzen für einzelne Abschnitte festlegt.

Code-Inspektion
: Der Abschnitt, in dem die →*Teilnehmer* erstmalig den Code des →*Ausgangssystems* untersuchen. Normalerweise lässt sich der →*Moderator* von den Teilnehmern durch den Code führen und erläutert Einzelaspekte auf Nachfrage oder auch von sich aus.

Ergebnissystem
: Die Software, wie sie in einer →*Teachlet-Einheit* von den →*Teilnehmern* aus dem →*Ausgangssystem* entwickelt wird. Das Ergebnissystem ist der →*Teachlet-Einheit* zuzuordnen und entsteht bei jeder Aufführung des →*Teachlets* erneut. Anders als das →*Zielsystem* ist das Ergebnissystem nicht in jedem Fall vollständig und fehlerfrei.

Lernziel
: Jedes →*Teachlet* hat ein Lernziel. Häufig ist das ein objektorientiertes Entwurfsmuster, es kommen aber auch z.B. Programmiersprachkonzepte und Umsetzungen von Algorithmen in Frage. Innerhalb einer →*Teachlet-Einheit* sollen die →*Teilnehmer* das Lernziel sowohl theoretisch erfahren als auch praktisch auf das →*Ausgangssystem* anwenden.

Moderator
Eine →*Teachlet-Einheit* wird von einem oder mehreren Moderatoren angeleitet. Diese müssen mit dem →*Teachlet* und seinem →*Lernziel* sehr gut vertraut sein, um die →*Teilnehmer* erfolgreich und innerhalb des zeitlichen Rahmens durch die →*Choreographie* zu lenken, ohne ihnen mehr Freiräume zu nehmen als nötig.

Teachlet
1. Ein auf Interaktivität ausgelegtes Lehrkonzept, das im vorigen Abschnitt sowie in Kapitel 4 genauer definiert wird.
2. Eine weitgehend statische Sammlung aller Dokumente und Software, die es ermöglichen, zu einem bestimmten Thema eine →*Teachlet-Einheit* zu halten, die vom →*Autor* bzw. von den Autoren dieses Teachlets konzipiert wurde. Es wird empfohlen[Sch05], für jedes Teachlet eine Versionsnummer zu vergeben, anhand der erkennbar ist, wie oft und wie stark dieses Teachlet bereits überarbeitet wurde.

Teachlet-Einheit
Eine konkrete Durchführung eines →*Teachlets* in einer bestimmten Lehreinheit.
Auch: *Teachlet-Aufführung*

Teachlet-Werkstatt
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.* “ ([Sch05], Verweis-Pfeile nicht im Original) In Teachlet-Werkstätten wird also der Schwerpunkt auf Feedback zu den neu erstellten Teachlets gelegt. Zu diesem Zweck werden dort meist auch →*Tracker* eingesetzt.

Teilnehmer
Zu einer →*Teachlet-Einheit* gehören mindestens einer, typischerweise mehrere Teilnehmer. Die Teilnehmer sitzen im Plenum und gestalten den vom →*Moderator* vorgegebenen Rahmen durch Wortmeldungen und Diskussionen. In der Regel lassen sich bei einer Teachlet-Einheit sowohl aktive als auch passive Teilnehmer beobachten. Aktive Teilnehmer äußern verstärkt ihre Meinung und vertreten ihre Position, während passive Teilnehmer weitgehend ausschließlich zuhören.

Tracker
Eine Rolle, der vor allem in der →*Teachlet-Werkstatt* eine wichtige Bedeutung zukommt. Der Tracker stammt aus dem Kreis der →*Teilnehmer* und hat die Aufgabe, ein Zeitprotokoll des →*Teachlets* zu erstellen, anhand dessen der →*Moderator* hinterher die Praxistauglichkeit der →*Choreographie* überprüfen kann. Dazu schreibt der Tracker mit, zu welcher Uhrzeit die verschiedenen Phasen des Teachlets beginnen und enden.

Zielsystem
Eine Version der betrachteten Software, in der das →*Lernziel* korrekt umgesetzt wur-

de. Das Zielsystem wird normalerweise vom →*Autor* des Teachlets erstellt und wird als Notlösung verwendet, falls die Implementierung den →*Teilnehmern* nicht innerhalb der Zeit gelingt, oder als Anschauungsmaterial für alternative Umsetzungen, falls sich die des Zielsystems von der der Teilnehmer unterscheidet.

2.3. Methodische Einordnung

Das Teachlet-Konzept existiert bisher losgelöst von der etablierten Begriffswelt der Didaktik. Aus diesem Grund soll es in diesem Abschnitt in das didaktische Modell nach Schulz et al. eingeordnet werden. In jenem Modell wird eine Unterscheidung von *Aktionsformen* (Handlungsmustern für Lehrende) und *Sozialformen* (Methoden für das Lernen in der Gruppe) getroffen[HOS97].

Als grundlegende Aktionsformen werden *Direktes Agieren* und *Indirektes Agieren* unterschieden. Beim direkten Agieren steht der Lehrende im Mittelpunkt; er stellt eine Frage, trägt Inhalte vor oder demonstriert einen Sachverhalt. Beim indirekten Agieren stellt der Lehrende Situationen her, „in denen er die Lernenden bewusst sich selbst überlässt“[HOS97], etwa durch eine vorbereitete Aufgabe. Direktes und indirektes Agieren sollen sich nicht gegenseitig ausschließen. Anzustreben ist eine zielführende Balance.

Ein Teachlet verlangt im Gegensatz zu einem Vortrag vom Moderator viel indirektes Agieren. Die Aktivierung der Teilnehmer und das sprichwörtliche „Abgeben der Zügel“ zieht sich von der Betrachtung des Ausgangssystems bis zur Implementierung der Veränderungen durch die ganze Einheit. Der Moderator ist in diesen Situationen keine Autoritätsfigur, sondern idealerweise lediglich ein verlängerter Arm für die Aktionen der Teilnehmer.

Die vier Grundstrukturen der Sozialformen sind *Frontalunterricht*, *Gruppenarbeit*, *Partnerarbeit* und *Einzelarbeit*. In diesem Spektrum fallen Teachlets komplett und eindeutig in den Bereich des Frontalunterrichts. Das ist daran erkennbar, dass zu jedem Zeitpunkt das Plenum als Ganzes angesprochen wird und immer nur eine aktive Informationsquelle existiert (bei der es sich nicht unbedingt immer um den Moderator handelt, sondern auch um Teilnehmer, die diskutieren oder das Geschehen anderweitig lenken). Bei den anderen drei Kategorien der Sozialformen gibt es mehrere parallel aktive Informationsquellen. Einzel- oder Partnerarbeit ist das auffälligste Merkmal, das Programmierübungen von Teachlets unterscheidet. Das methodische Ziel von Teachlets, die Teilnehmer an einer gemeinsamen Sicht alle zusammen arbeiten zu lassen, ordnet sie im Sinne von Schulz et al. klar als Frontalunterricht ein.

Einige Abschnitte eines Teachlets zeigen Charakteristika verschiedener Varianten des *Unterrichtsgesprächs*[Bit06]. Je nach Stil des Moderators und des Teachlets gibt es *Diskussionen*, *gelenkte Gespräche* oder *freie Gespräche*.

3. Berichte von Moderatoren

Um Aussagen darüber treffen zu können, wofür und wie gut Teachlets funktionieren, wurden drei erfahrene Teachlet-Moderatoren nach ihren Erlebnissen und Einschätzungen befragt: Axel Schmolitzky, Carola Lilienthal und Christian Späh wurden interviewt. Alle drei haben Teachlets mehrfach selbst in Lehrveranstaltungen eingesetzt und haben hinreichend viel Erfahrung, um Stärken und Schwächen des Konzepts einschätzen zu können. Details zu ihrem jeweiligen Erfahrungsschatz und bisherigen Berührungspunkten mit Teachlets lassen sich in den Protokollen der Interviews im Anhang nachlesen.

Um der Individualität der jeweiligen Erfahrungen Rechnung tragen zu können, wurden Leitfadeninterviews[BMM06] durchgeführt. Bei dieser Methode der qualitativen Sozialforschung werden Befragungen anhand eines Leitfadens durchgeführt, der konkrete Fragen und Themen enthält. Im Einzelfall kann und soll von diesem allerdings abgewichen werden, wenn dadurch die Erfassung der Perspektive der interviewten Person gefördert wird. Zu Lasten der unmittelbaren Vergleichbarkeit der Interviews wird der Fragenkatalog nicht starr abgearbeitet, sondern im Verlauf des Interviews verändert und ergänzt.

Der Interviewleitfaden für die drei Interviews bestand aus den folgenden Fragen:

1. Welche Erfahrungen mit dem Teachlet-Konzept haben Sie bereits gesammelt?
2. Wie haben Sie Teachlets erlebt im Vergleich mit anderen Lehrformen, die Sie kennen?
3. Welche Vorteile sehen Sie an ihnen? Wo lassen sich Teachlets sehr gut einsetzen?
4. Wo sehen Sie Grenzen des Konzepts? Wo würden Sie Teachlets auf keinen Fall einsetzen?
5. Haben Sie Hinweise für zukünftige Teachlet-Moderatoren? Was ist noch kein etabliertes Wissen, sollte aber auf jeden Fall bedacht werden?

Die Protokolle der Interviews finden sich im Anhang ab Seite 32.

Zusätzlich liegt von Kai Meyer, dem Moderator eines Teachlets vor einer ungewöhnlich hohen Teilnehmerzahl, ein schriftlicher Bericht vor, der ebenfalls dem Anhang beigefügt ist (Seite 56).

3.1. Ergebnisse

Die Aussagen der interviewten Moderatoren erlauben es, die Praxistauglichkeit von Teachlets zu bewerten und Vor- und Nachteile sowie Stolperfallen zu benennen. Hier sind die zusammengefassten wichtigen Punkte:

- Teachlets sind ein gutes Werkzeug, um abstrakte Zusammenhänge anhand konkreter Probleme zu vermitteln.
- Dadurch, dass die Teilnehmer aktiv sind und das Geschehen steuern, setzen sie sich sehr viel intensiver mit den Inhalten auseinander.

- Durch die Interaktion mit den Teilnehmern gibt es nicht selten auch einen Lernprozess für den Moderator.
- Teachlets sind als Lehrform in der Durchführung anspruchsvoller als gewöhnliche Vorträge und erfordern sorgfältige Vorbereitung sogar für wiederholte Aufführungen des gleichen Teachlets durch den gleichen Moderator.
- Teachlets mit hohen Teilnehmerzahlen haben ihre eigenen speziellen Schwierigkeiten und sind kaum erprobt, bieten jedoch potenziell noch viel Raum für Erfolge.
- Es ist leicht, die Vorbereitungszeit zu unterschätzen. Vielen Moderatoren, die Teachlets zum ersten Mal durchführen, passiert genau das.
- Bei einer Teachlet-Einheit sind i.d.R. nicht alle Teilnehmer aktiv. Im Gegenteil gibt es meist nur einen vergleichsweise kleinen Anteil an Teilnehmern, die sich gesteigert an Diskussionen beteiligen und Wortmeldungen abgeben. Die eher passiven Teilnehmer haben trotzdem die Chance, durch die wechselnden Perspektiven im Diskurs mehr zu lernen und besser zu folgen als bei einem einfachen Vortrag.
- Selbst, wenn ein Teachlet von der Moderation her nicht optimal läuft, kann es für die Teilnehmer sehr lohnenswert sein. Bis zu einem gewissen Grad können die Teilnehmer eine mangelhafte Moderation abfedern, was bei einem Vortrag überhaupt nicht möglich ist.
- Teachlets könnten noch in viel mehr Bereichen eingesetzt werden, als sie es bisher wurden.

Alle befragten Moderatoren bewerten Teachlets als überaus positiv und empfehlen die Methode weiter.

4. Definition

Seit 2005 existiert eine Definition für den Teachlet-Begriff[Sch05]. Diese Definition hat die wichtige Aufgabe erfüllt, das Konzept zu umreißen und eine erste feste Vorstellung von Teachlets zu etablieren. Seitdem sind sie vielfältig eingesetzt worden und es ist an der Zeit, die Definition auf ihre Grenzen und Probleme zu untersuchen. Im zweiten Unterabschnitt dieses Kapitels wird eine aktualisierte Definition konstruiert, die die Entwicklungen der Praxis berücksichtigt und Schwachstellen der alten Definition behebt.

4.1. Die ursprüngliche Definition und ihre Grenzen

Die erste Teachlet-Definition[Sch05] in vollständiger Form:

> *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.*

Vor dem Hintergrund der erlebten Praxis der letzten Jahre fallen an dieser Definition mehrere Aspekte auf, die verbesserungswürdig sind:

- Die *Interaktivität* sollte etwas mehr hervorgehoben werden. Teachlets einfach nur als „interaktiv“ zu bezeichnen, wird ihnen nicht gerecht, da auch Lehrkonzepte wie Vorträge mit Frage-und-Antwort-Abschnitten als interaktiv zu bezeichnen wären. Bei Teachlets ist die Interaktivität jedoch kein beobachteter, zufälliger Effekt, sondern zentrales, erklärtes Ziel.
- Die *Erweiterung der Funktionalität* ist eine unnötige Einschränkung möglicher Aufgabenstellungen. In einem Teachlet könnte stattdessen etwa auch ein Refactoring durchgeführt werden, um ein Entwurfsmuster zu veranschaulichen. In diesem Sinne sollte auch etwas weiter unten im Text statt von einer vorzunehmenden „Erweiterung“ eher von einer vorzunehmenden *Veränderung* die Rede sein.
- Die Formulierung „ein Entwurfsmuster oder ein Programmiersprachkonzept“ ist ebenfalls recht restriktiv und sollte im Hinblick auf andere mögliche Lernziele (bestimmte Algorithmen, Refactorings o.Ä.) verallgemeinert werden.
- Es ist festgelegt, dass die Entwicklung „mit Hilfe (...) eines Beamers“ stattfinden muss. Diese Einschränkung deckt zwar fast alle bisherigen Teachlets ab, berücksichtigt jedoch nicht die technologische Entwicklung und die mögliche (wenn auch noch unerprobte) Option von Teachlets per Telekommunikation, z.B. per Videokonferenz. Entscheidend ist konzeptuell lediglich eine *gemeinsame Sicht* auf die Software, im Unterschied zum Szenario, in dem jeder Teilnehmer für sich allein entwickelt.

- Letztlich ist in Frage zu stellen, inwieweit Teachlets auf die Veränderung von Quelltext festgelegt sein sollten, wo es doch auch z.B. visuelle Programmierung und weitere Paradigmen der Softwareentwicklung gibt, in denen der Begriff des Quelltextes zugunsten anderer Implementationsmechanismen wegfällt.

Diese Teachlet-Definition ist somit an einigen Stellen zu restriktiv formuliert. Dennoch sollen auch die ausdrücklich positiven Aspekte hervorgehoben werden: Gut ist, dass die Definition festschreibt, dass es sich bei dem Untersuchungsgegenstand um „lauffähige Software“ handelt, denn die Programmierung von Software ist die Domäne, in der Teachlets eingesetzt werden und für die sie konzipiert sind. Weiterhin ist es sinnvoll, dass Moderator und Teilnehmer genannt werden und dass hervorgehoben wird, durch welche Aufgaben sich diese Rollen auszeichnen. Diese Aspekte gilt es, bei der Überarbeitung der Definition in ihrer Deutlichkeit zu erhalten.

4.2. Aktualisierte Definition

Mit Hilfe der Kritikpunkte aus dem vorigen Kapitel ist es nun das Ziel, eine neue Teachlet-Definition aufzustellen. Der erste Schritt zu einer neuen Definition ist es, alle wichtigen Aspekte so vollständig wie möglich aufzuzählen.

Zu diesem Zweck wurde bei einem Treffen mit Axel Schmolitzky, dem Erfinder des Konzepts, und Christian Späh, einem Teachlet-Moderator der ersten Stunde, eine Sammlung wichtiger Punkte erstellt und strukturiert (siehe Abbildung 4).

Dabei ist die folgende Liste von essenziellen Teilaspekten zusammengekommen:

- Für das Teachlet:
  - Aufgabe
  - Lernziel
  - Lösungsraum
  - Ausgangssystem
  - Implementation
  - Architektur
  - Zielsystem
  - Dokumentation
- Für die Teachlet-Einheit:
  - Rechner
  - Standard-Ablauf
  - Gemeinsame Sicht (z.B. Beamer)
  - Moderator
  - Teilnehmer (aktive, ≥ 1)

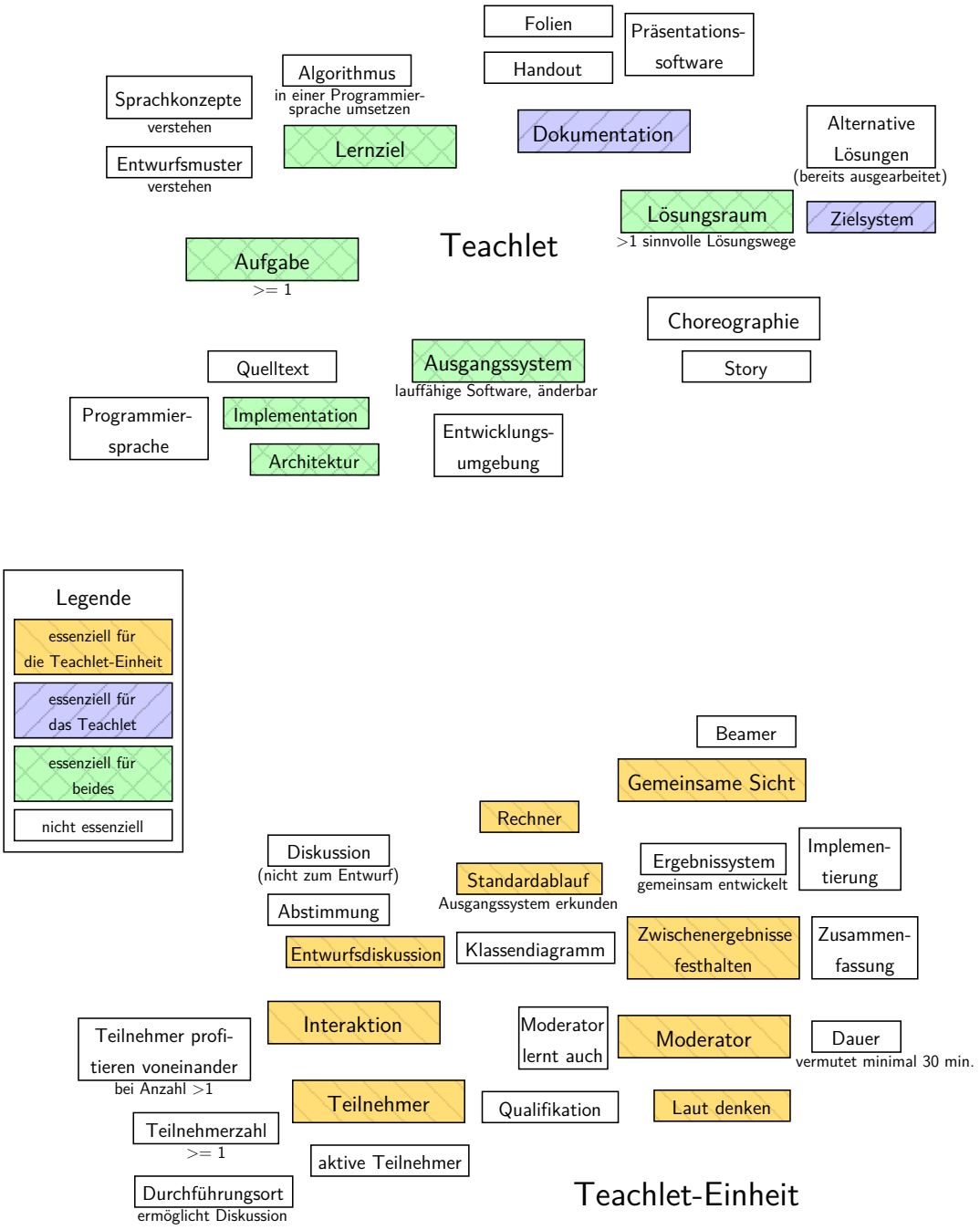


Abbildung 4: Diese Übersicht über die wichtigen Aspekte von Teachlets und Teachlet-Einheiten wurde von mehreren erfahrenen Teachlet-Moderatoren gemeinsam entwickelt. Verbindungen zwischen den Begriffen sind nicht explizit visualisiert, semantische Nähe wird durch räumliche Nähe dargestellt.

- Zwischenergebnisse festhalten
- Entwurfsdiskussion
- Interaktion

Die restlichen Begriffe aus Abbildung 4 wurden als wichtig, aber nicht unverzichtbar gewertet.

Mit dieser Liste zentraler Aspekte und den Kritikpunkten aus dem vorigen Abschnitt ist es jetzt möglich, schrittweise eine neue Teachlet-Definition aufzustellen. Dafür wird die alte Definition Satz für Satz überarbeitet und danach weitere Punkte angefügt. Der erste Satz der alten Definition:

> *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.*

Ausgehend von den ersten drei Kritikpunkten der obigen Liste ergibt sich der folgende überarbeitete Satz:

> *Ein Teachlet ist eine sehr interaktive Lehreinheit, in der ein lauffähiges Stück Software so verändert wird, dass es eine klar definierte Aufgabenstellung erfüllt. Diese ist so gewählt, dass durch die Veränderung ein klar definiertes Lernziel erreicht werden kann.*

Danach folgt der zweite Satz:

> *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 verbleibenden Kritikpunkte aufgreifend wird der Satz wie folgt überarbeitet:

> *Ein Moderator motiviert das Ausgangssystem sowie die vorzunehmende Veränderung in einer mit den Teilnehmern gemeinsamen Sicht (etwa am Beamer) und lässt sich dann von den Teilnehmern anleiten, die dazu notwendigen Änderungsschritte an der Software vorzunehmen.*

Die Original-Definition ist hiermit bereits beendet. Aus der obigen Liste der essenziellen Begriffe sind die folgenden noch nicht genannt: *Lösungsraum*, *Implementation*, *Architektur*, *Zielsystem*, *Dokumentation*; *Rechner*, *Standard-Ablauf*, *Zwischenergebnisse festhalten*, *Entwurfsdiskussion*.

Zwei davon sollen begründet ausgeklammert werden: *Rechner* weil das Vorhandensein eines Rechners durch die Veränderung lauffähiger Software implizit ist, und *Standard-Ablauf* weil dieser gerade durch die Definition dargestellt werden soll.

Um die restlichen Begriffe abzudecken, werden ein paar weitere Sätze angefügt:

Für die Lösung der Aufgabe soll ein Lösungsraum mit mehreren möglichen Varianten hinsichtlich der Architektur existieren, um eine Entwurfsdiskussion zu ermöglichen. Der Moderator hat dabei die Aufgabe, die Diskussion zu lenken, Zwischenergebnisse festzuhalten und eine Einigung über eine Zielsetzung für die Implementation herbeizuführen. Nach der gemeinsamen Implementierungsphase erfüllt die Software idealerweise die zuvor gestellte Aufgabe.

Zu einem Teachlet gehört weiterhin eine Dokumentation, die für die Vorbereitung relevante Unterlagen wie eine detaillierte Choreographie und ein ausimplementiertes Zielsystem enthält.

Insgesamt ergibt sich also die folgende aktualisierte Teachlet-Definition:

Ein Teachlet ist eine sehr interaktive Lehreinheit, in der ein lauffähiges Stück Software so verändert wird, dass es eine klar definierte Aufgabenstellung erfüllt. Diese ist so gewählt, dass durch die Veränderung ein klar definiertes Lernziel erreicht werden kann.

Ein Moderator motiviert das Ausgangssystem sowie die vorzunehmende Veränderung in einer mit den Teilnehmern gemeinsamen Sicht (etwa am Beamer) und lässt sich dann von den Teilnehmern anleiten, die dazu notwendigen Änderungsschritte an der Software vorzunehmen.

Für die Lösung der Aufgabe soll ein Lösungsraum mit mehreren möglichen Varianten hinsichtlich der Architektur existieren, um eine Entwurfsdiskussion zu ermöglichen. Der Moderator hat dabei die Aufgabe, die Diskussion zu lenken, Zwischenergebnisse festzuhalten und eine Einigung über eine Zielsetzung für die Implementation herbeizuführen. Nach der gemeinsamen Implementierungsphase erfüllt die Software idealerweise die zuvor gestellte Aufgabe.

Zu einem Teachlet gehört weiterhin eine Dokumentation, die für die Vorbereitung relevante Unterlagen wie eine detaillierte Choreographie und ein ausimplementiertes Zielsystem enthält.

An dieser neuen Definition fällt sofort auf, dass sie deutlich länger ist als die alte. Dabei sollte kritisch betrachtet werden, ob eine längere Definition den Begriff in diesem Fall tatsächlich präziser beschreibt oder ob der gleiche Informationsgehalt auch in einem deutlich kürzeren Text vermittelt werden kann.

Setzt man die Prägnanz einer Definition als Qualitätsmaßstab an, so ergibt sich aus dieser Sicht, dass sie mit möglichst wenigen, wohlüberlegten Worten möglichst viele zentrale Informationen vermitteln sollte. Obwohl eine detaillierte sprachliche Analyse der Definition nicht im Fokus dieser Arbeit liegt (und im Sinne der wissenschaftlichen Unbefangenheit wohl auch besser von jemand anderem als dem Autor durchgeführt werden sollte), drängt sich die Möglichkeit auf, die bereits identifizierten zentralen Begriffe als zählbare Informationseinheiten zu betrachten. Auf diese Weise lässt sich wenigstens ein grober Vergleich der alten und neuen Definition hinsichtlich des „Informationsgehalts“ bezogen auf die zentralen Teachlet-Aspekte vornehmen.

In der alten Definition finden sich von den zentralen Begriffen etwa acht bis zehn, je nachdem wie eng ähnliche Formulierungen und verwandte Ideen gesehen werden. Dem gegenüber steht die neue Definition, die bei einer etwa doppelten Länge dafür auch alle 16 Kernbegriffe (*Teachlet* und *Teachlet-Einheit* eingerechnet) enthält. Darauf begründet sich die Behauptung, dass die neue Definition in Sachen Prägnanz kein Rückschritt ist, sondern dass die größere textuelle Länge mit einem proportionalen inhaltlichen Zuwachs einhergeht, der nicht nur sinnvoll, sondern nötig ist.

5. Möglichkeiten und Grenzen

Um die neue Definition genauer zu untersuchen und zu erläutern, sollen jetzt ihre einzelnen wichtigen Aspekte dahingehend überprüft werden, wie es sich auswirkt, sie abzuschwächen oder komplett zu entfernen. Wir werden sehen, dass es einige Varianten von Teachlets gibt, die noch nicht in der Praxis erprobt worden sind und evtl. Potenzial für weitere Erfolge bereithalten.

5.1. Das Durchschnitts-Teachlet

Die größte Konzentration von Teachlets, nämlich etwa ein Dutzend innerhalb eines Semesters, findet sich in der Teachlet-Werkstatt an der Universität Hamburg. Darüber hinaus werden Teachlets nur vereinzelt und verstreut eingesetzt. Von den speziellen Konzepten der Teachlet-Werkstatt (Tracker, Feedbackformulare) abgesehen, hat das durchschnittliche Teachlet des Jahres 2010 also in etwa folgende Eigenschaften:

- eine Länge von ca. 90 Minuten
- zwischen 10 und 15 Teilnehmer
- ein bis zwei Moderatoren
- einen Seminarraum als Durchführungsort mit einem Beamer als gemeinsame Sicht
- Präsentationsfolien
- Verwendung einer Kreidetafel oder eines Whiteboards zum Live-Erstellen von Diagrammen (insbesondere Klassendiagrammen)
- ein Ausgangssystem in Java, Programmierung in Eclipse
- ein objektorientiertes Entwurfsmuster als Lernziel

Die Vorgehensweise, an einer Tafel gemeinsam Klassendiagramme zu erstellen, ist nicht in der Definition gefordert und keine Bedingung für ein erfolgreiches Teachlet, sie hat sich jedoch so gut bewährt, dass sie in fast jedem Teachlet der Teachlet-Werkstatt eingesetzt wird. Dabei zeichnet der Moderator nach Anweisungen der Teilnehmer nach der ersten Code-Inspektion ein Klassendiagramm des Ausgangssystems an, das alle für die Umsetzung relevanten Infos zu Klassen, Beziehungen und Schnittstellen enthält, damit allen die Architektur des Systems klar wird. Nach der Entwurfsdiskussion kann dieses Klassendiagramm an die geplante Umsetzung angepasst werden, so dass alle eine recht genaue Zielvorstellung für die Implementation haben.

Von diesen Eigenschaften können die Teachlets allerdings auch in einem oder mehreren Aspekten stark abweichen. Im Abschnitt 5.3 wird u.A. erkennbar, wie gut das im Einzelnen funktioniert hat. Vorher sollen jedoch ein paar Richtlinien dafür gegeben werden, wann eine Lehreinheit die Grenzen des Konzepts überschreitet und definitiv kein Teachlet mehr ist.

5.2. Ausschlusskriterien und Grenzen

Auf die Liste essenzieller Aspekte aus Kapitel 4 besteht an dieser Stelle ein erneuter Bezug. Hier werden die Listen für Teachlet und Teachlet-Einheit zusammengefasst. Jeder der Punkte soll dann dahingehend bewertet werden, inwieweit er für das Konzept in jeder Hinsicht unverzichtbar ist und ob ein Spielraum besteht. Die Begriffe werden in einer sinnvollen Reihenfolge abgearbeitet.

Moderator: *unverzichtbar*
Für die Durchführung eines Teachlets müssen die Aufgaben des Moderators in jedem Fall erledigt werden. Dafür muss es unbestreitbar wenigstens einen Moderator geben. Falls kein designierter Moderator existiert, müsste die Rolle von einzelnen Teilnehmern ausgeführt werden. Im Sinne einer reibungslosen Durchführung muss jedoch mindestens ein Moderator existieren, der selbst kein Teilnehmer ist. Streitbar ist höchstens die maximale Anzahl, bevor die Moderation nicht mehr effizient erfolgen kann. Das sollte jedoch vom Einzelfall abhängig sein und keine feste Grenze erfordern.

Teilnehmer: *unverzichtbar*
Damit man von einem Teachlet sprechen kann, muss ein Lern- und Lehrprozess stattfinden. Dazu muss es Teilnehmer geben, die den Verlauf aktiv beeinflussen. Wenn es nur passive oder überhaupt keine Teilnehmer gibt, kann keine Interaktion stattfinden. Das Teachlet degeneriert dann bestenfalls zum Vortrag oder kann schlimmstenfalls überhaupt nicht durchgeführt werden.

Interaktion: *unverzichtbar*
Zum Teachlet-Konzept gehört die Interaktion zwischen Moderator und Teilnehmern fundamental dazu. Ohne Interaktion liegt lediglich ein Vortrag und definitiv kein Teachlet vor. Dennoch gibt es Spielraum dabei, wie stark diese Interaktion ausgeprägt ist und welche Formen sie annimmt. Unter der Bedingung, dass aktive Teilnehmer existieren, kann bspw. auf ausführliche Diskussionen verzichtet werden, wenn dafür andere Mechanismen verwendet werden, um die Teilnehmer einzubeziehen. Möglich wären dafür etwa kurze Zurufe oder Abstimmungen per Handzeichen. Entscheidend ist, dass die Teilnehmer den Entwurf bestimmen und die Richtung angeben.

Standard-Ablauf: *eingeschränkt verzichtbar*
Der Standard-Ablauf eines Teachlets, also die grobe Richtung der meisten Choreographien, besagt, dass als erstes die Motivation und Betrachtung des Ausgangssystems erfolgt, dann die Aufgabenstellung, die Entwurfsdiskussion, die Umsetzung und am Ende eine Zusammenfassung. An dieser Reihenfolge etwas maßgeblich zu verändern würde schon rein logisch keinen Sinn ergeben, da das Ausgangssystem bekannt sein muss um die Aufgabe zu verstehen, die Aufgabe geklärt sein muss um über die Möglichkeiten der Umsetzung diskutieren zu können, und die Möglichkeiten der Umsetzung gefunden sein müssen, bevor man sie durchführen kann. Eine Zusammenfassung am Ende bietet sich stark an.

Allerdings kann man einige Schritte in geringem Maße schieben. So hat es zum Beispiel bereits Teachlets gegeben, in denen die Betrachtung des Ausgangssystems aufgeteilt

wurde, so dass erst die Funktionalität und Verwendung gezeigt, dann die Aufgabe besprochen und danach erst das erste mal der Quelltext erkundet wurde. Der Standard-Ablauf ist eine grobe Schablone, in die verschiedene konkrete Ausgestaltungen passen können.

Aufgabe: *unverzichtbar*
Es ist ein Kernbestandteil eines Teachlets, dass es eine Aufgabe gibt, die gemeinsam bewältigt werden soll. Ob diese Aufgabe in einer Teachlet-Einheit komplett erledigt wird, ist eine Frage der jeweiligen Durchführung. In jedem Fall muss sie jedoch vorhanden und klar gestellt sein, um sicherzustellen, dass alle Teilnehmer am gleichen Problem arbeiten.

Lernziel: *unverzichtbar*
Bei einem Teachlet geht es letztlich darum, den Teilnehmern eine Chance zu geben, etwas zu lernen. Ein Teachlet ohne Lernziel würde schon den Begriff, der ja vom englischen *teach*, also *lernen*, kommt, ad absurdum führen. Es wäre methodisch interessant, was passiert, wenn man das Lernziel nicht im Vorhinein festlegt, sondern eine lauffähige Software explorativ mit den Teilnehmern untersucht. Als Teachlet wäre das dann aber wohl nicht mehr zu bezeichnen.

Ein bekannter Effekt der Teachlet-Methode ist, dass auch der Moderator eines Teachlets in der Durchführung von den Teilnehmern etwas lernen kann. Das passiert nicht immer, aber durch die starke Interaktion ist es ein mögliches Szenario.

Lösungsraum: *unverzichtbar*
Es ist offensichtlich, dass die Aufgabe eines Teachlets lösbar sein muss. Für ein gelungenes Teachlet hat sie ein Niveau, auf dem sie von den Teilnehmern gelöst werden kann, ohne trivial zu erscheinen.

Weiterhin soll der Lösungsraum, also die Menge aller möglichen Lösungen, größer sein als eins. Wenn es zu einem Problem nur eine einzige, offensichtliche Lösung gibt, dann ist keine Diskussion möglich. In einem solchen Fall wird von einem oder mehreren Teilnehmern die Lösung vorgeschlagen und dann umgesetzt, weil es keine abweichenden Vorschläge gibt – es kommt kein Gespräch zustande. Hierbei geht es nicht darum, dass die Teilnehmer keine Ahnung haben, in welche Richtung sie denken sollen, im Gegenteil kann es je nach Lernziel sinnvoll sein, vor der Diskussion ein passendes Entwurfsmuster abstrakt vorzustellen. Worauf es jedoch ankommt, ist, dass die Teilnehmer mehrere Möglichkeiten der Umsetzung erkennen und Vor- und Nachteile abwägen können.

Entwurfsdiskussion: *eingeschränkt verzichtbar*
Wenn ein nichttrivialer Lösungsraum gegeben ist, dann kann die Entwurfsdiskussion wie von selbst entstehen. Das Ziel dabei ist, dass sich die Teilnehmer einigen, wie das vorliegende Problem in der Software gelöst werden kann. Diese Diskussion sollte der Moderator ihnen in keinem Fall verweigern (auch wenn in einigen Fällen eine stringente Moderation und ggf. sogar ein Abbruch aus Zeitgründen nötig sein kann). Anders formuliert: Die Frage nach den Umsetzungsmöglichkeiten sollte in jedem Fall gestellt werden.

Dennoch kann es sich in der Durchführung ergeben, dass die Teilnehmer nur eine einzige Lösung sehen oder sich sofort einig sind. In solchen Fällen kann der Moderator noch weitere Lösungen zur Diskussion stellen. Falls das nicht möglich ist, etwa aus Zeitgrün-

den oder weil die Kenntnisse der Teilnehmer nur die eine Lösung ermöglichen, kann die Entwurfsdiskussion auch sehr kleinschrittig durchgeführt oder vorzeitig beendet werden.

In einem solchen Fall, wenn also eine Entwurfsdiskussion nicht möglich ist, kann sie auch nicht erzwungen werden. Der Teachlet-Grundgedanke ist dann immer noch erfüllt – anders als in dem Fall, in dem die Entwurfsdiskussion bewusst unterdrückt wird.

Architektur: *unverzichtbar*
Wenn es in der Softwareentwicklung um Entwurfsentscheidungen geht, spielt meist die Architektur der Software eine Rolle. Die Möglichkeiten der Architektur stellen dann den Lösungsraum des Teachlets. Dabei muss die Architektur nicht unbedingt objektorientierter Art sein – bei anderen Paradigmen der Programmierung spielt die Strukturierung der Software ebenso eine wichtige Rolle. Entsprechend steht die Diskussion um Architektur und Entwurf bei vielen Teachlets im Mittelpunkt.

Zwischenergebnisse festhalten: *eingeschränkt verzichtbar*
Bei einem Teachlet wie auch bei einem klassischen Vortrag ist es für die Teilnehmer eine große Unterstützung, wenn kontinuierlich Zusammenfassungen dessen vorgebracht werden, was bisher passiert ist. In gesteigertem Maße gilt das für die Entwurfsdiskussion in einem Teachlet, in der mehrere Ansätze parallel diskutiert und evaluiert werden müssen. Dabei ist der Moderator gefordert, das Gesagte immer im Ganzen präsent zu halten, zusammenzufassen und auf das Ziel zu orientieren. Ohne diese Maßnahmen kann ein Teachlet sowohl aus dem zeitlichen als auch aus dem inhaltlichen Rahmen laufen.

Rechner: *unverzichtbar*
Da in einem Teachlet an lauffähiger Software live gearbeitet wird, kann es ohne Rechner nicht durchgeführt werden. In der Regel gibt es dafür einen Präsentationsrechner und die Teilnehmer haben keine eigenen Rechner.

Gemeinsame Sicht: *unverzichtbar*
Der größte Unterschied zwischen einem Teachlet und einer Programmierübung, in der jeder für sich arbeitet, ist, dass die Teilnehmer eines Teachlets gemeinsam an einer Instanz der Software arbeiten. Dies ergibt sich schon notwendigerweise daraus, dass i.A. nur der Moderator einen Rechner verwendet. Die Teilnehmer können deshalb keine Änderungen für sich allein ausprobieren, sondern sind sozusagen gezwungen, ihre Ideen vor der Umsetzung mit den restlichen Teilnehmern zu besprechen und zu diskutieren. Der Moderator kann dazu ermutigen, mit verschiedenen Varianten zu experimentieren, aber dennoch entsteht nicht die Situation, dass Teilnehmer völlig unüberlegt und unbegründet irgendetwas ausprobieren.

Die gemeinsame Sicht muss nicht unbedingt physisch existieren. Der Normalfall ist zwar ein Beamer am Präsentationsrechner, aber unter Umständen könnten die Teilnehmer das Geschehen auch auf einem anderen, ggf. entfernten Bildschirm verfolgen, solange gewährleistet bleibt, dass alle das gleiche sehen.

Ausgangssystem: *eingeschränkt verzichtbar*
Notwendig für ein Teachlet ist die Arbeit mit lauffähiger Software. In der bisherigen Praxis war diese Software immer bereits im Vorfeld als Ausgangssystem gegeben und

sollte in einer bestimmten Weise verändert werden. Das ist nur unter ganz bestimmten Umständen anders denkbar und wurde noch nicht praktisch erprobt (siehe Abschnitt 5.3).

Implementation: *eingeschränkt verzichtbar*
Die Umsetzung des Lernziels am Ausgangssystem, also die erfolgreiche Anpassung der Software, sollte das Ziel jeder Teachlet-Planung sein. In der Praxis kommt es allerdings vor, dass die Zeit für eine vollständige Implementierung nicht ausreicht. In einem solchen Fall ist das Fingerspitzengefühl des Moderators gefragt: Es sollte zumindest so ausführlich programmiert werden, dass die Umsetzung für die Teilnehmer komplett und mit allen Implementationsdetails vorstellbar ist. Wenn nur noch etwas Code eingegeben werden muss, aber allen klar ist, was genau noch fehlt, kann stattdessen auch die Implementierung abgebrochen und ein vorbereitetes Zielsystem gezeigt werden.

Zielsystem: *eingeschränkt verzichtbar*
Ein vorbereitetes Zielsystem zu erstellen ist sehr empfehlenswert, da es beim Teachlet aus Zeitmangel oder zum Vergleich von Entwürfen notwendig werden kann. Bei der Vorbereitung des Teachlets durch den Moderator entsteht das Zielsystem schon allein dadurch, dass es sich für den Moderator empfiehlt, die Programmieraufgabe im Vorfeld mindestens einmal komplett selbst zu lösen. Bei den meisten vorhandenen Teachlets sollte ein Zielsystem bereits dabei sein, da es zu den Vollständigkeits-Ansprüchen der Teachlet-Werkstatt gehört.

Es ist an dieser Stelle als eingeschränkt verzichtbar bewertet, weil es nicht zum Einsatz kommen muss, wenn das Teachlet perfekt läuft. Dennoch ist es sehr empfehlenswert, ein Zielsystem vorbereitet zu haben, um die Tragweite von unvorhergesehenen Problemen einzuschränken.

Dokumentation: *verzichtbar*
Die Erstellung einer vollständigen Dokumentation ist unerlässlich, um das Teachlet für die Nachwelt zu erhalten. Davon unabhängig ist ein Teachlet auch völlig ohne Dokumentation potenziell durchführbar und kann gut funktionieren, weshalb die Dokumentation als einziger Begriff in diesem Abschnitt als verzichtbar bewertet wird. Dennoch ist es in keiner Weise empfehlenswert, auf die Erstellung von Dokumentation zu verzichten. Zumindest ein vorbereitetes Zielsystem und eine ausgearbeitete Choreographie tragen maßgeblich dazu bei, dass ein Teachlet gelingen kann und dass der Moderator es souverän durchführen kann.

5.3. Varianten und Möglichkeiten

In diesem Abschnitt sollen die bisherigen Gedanken und Erkenntnisse genutzt werden, um einige vielversprechende Varianten des Teachlet-Konzepts zu beschreiben, die so noch nicht oder nur sehr selten umgesetzt worden sind. Es handelt sich hierbei um eine Liste von Beispielen, die keinen Anspruch auf Vollständigkeit erhebt. Sie soll dazu ermutigen, diese neuen Facetten des Konzepts praktisch zu erproben und versucht, den Personen, die ein solches Teachlet vorbereiten, wichtige Hinweise zu geben, damit es gelingen kann.

5.3.1. Teachlets mit vielen Teilnehmern

Teachlets sind bisher zumeist in seminarartigen Kontexten wie der Teachlet-Werkstatt durchgeführt worden. Eine Frage, die sich aufdrängt, ist, mit wie vielen Teilnehmern ein Teachlet funktionieren kann.

Zur Minimalanzahl der Teilnehmer wurde weiter oben bereits eine Aussage getroffen: Nötig ist mindestens ein Teilnehmer, damit eine Interaktivität und eine Diskussionskultur entstehen kann.

Die Obergrenze, sofern es eine gibt, ist deutlich schwieriger zu fassen. Vieles entscheidet sich bei näherer Betrachtung mehr durch die Eigenschaften des Durchführungsortes als durch die Anzahl der Teilnehmer an sich. Ob in einem Seminarraum 10 oder 40 Teilnehmer sitzen, ist für den Moderator weitgehend unerheblich. Entscheidend ist dort, dass der Moderator und alle Teilnehmer ohne akustische Verstärkung sprechen und sich gegenseitig verstehen können. Das erlaubt mündliche Diskussionen und einen regen Austausch von Ideen.

Wenn die Anzahl der Teilnehmer an und über 50 steigt, ist der Durchführungsort vermutlich eher ein Vorlesungssaal oder etwas Vergleichbares. In dem Maße, in dem der Austausch untereinander schwieriger wird, muss der Moderator die Interaktion mit den Teilnehmern kleinschrittiger gestalten und Fragen stellen, die eine Antwort per kurzem Zuruf auch aus den hinteren Sitzreihen erlauben. Das schließt die Besprechung von Entwurfsfragen keineswegs aus, rückt allerdings die kleinen Schritte in den Vordergrund. Ein Beispiel: Statt nach Vorschlägen für eine Klassenstruktur zu fragen, die langen Erläuterungen bedürften, kann der Moderator eher nach der Benennung einer Klasse fragen oder von welcher bestehenden Klasse sie erben soll. Noch deutlich offener ist die mögliche Frage: „Was ist der nächste Schritt?“ Zur eigenen Orientierung sollte der Moderator nur Fragen einplanen, die er selbst in einem oder zwei Sätzen beantworten könnte.

Die Erfahrung zeigt, dass mit der fünffachen Teilnehmerzahl keine Verfünffachung der Anzahl der Wortmeldungen zu befürchten ist. Auch bei vielen Teilnehmern ist es nur ein kleiner Teil – erfahrene Teachlet-Moderatoren sprechen von 5 bis 10 – die sich aktiv beteiligen. Das bedeutet nicht, dass alle anderen unaufmerksam wären oder nichts lernen würden.

5.3.2. Teachlets ohne physische Präsenz

Ein bisher unerprobter Gedanke ist die Durchführung von Teachlets ohne physische Anwesenheit der Teilnehmer. Dies wäre zum Beispiel denkbar, indem eine Software ver-

wendet wird, die Bild und Ton zwischen Moderator und Teilnehmern übertragen kann. Wichtig sind hier beide Richtungen, da Input auch von den Teilnehmern kommen können muss. Eine Kombination aus Audiokonferenz und Videostream würde die nötigen Kommunikationskanäle schaffen, um ein solches Teachlet durchführen zu können, das trotz indirekter Kommunikation die Definition und den Sinn eines Teachlets erfüllt.

Mangels Erfahrung mit der erforderlichen Technik kann der Autor leider keine weiteren Hinweise geben, was speziell auf Teachlets bezogen bei der Durchführung beachtet werden muss.

Für „virtuelle Vorlesungen“ existiert eine ganze Reihe von Systemen[CDS09], die natürlich noch im Einzelnen auf ihre Eignung überprüft werden müssten.

5.3.3. Teachlets ohne Ausgangssystem

Die Teachlet-Definition aus Kapitel 4 spricht von Veränderung lauffähiger Software als Kriterium für ein Teachlet. Als Formulierung ist das sicher zweckmäßig, da in bisherigen Teachlets in aller Regel eine existierende Software erweitert oder umgebaut wurde. Dabei stellt sich die Frage, ob man in einem Teachlet auch eine Software von Grund auf mit den Teilnehmern erstellen könnte.

Das erscheint grundsätzlich vorstellbar. Es dürfte auf die verfügbare Zeit und auf das Lernziel ankommen, ob diese Vorgehensweise sinnvoll ist. Vermutlich müssen die Teilnehmer sehr gut mit der Entwicklungsumgebung und der Programmiersprache vertraut sein.

Ein Vorteil dieses Ansatzes ist, dass keine Teilnehmer bereits beim Betrachten des Ausgangssystems überfordert werden können – die Gefahr kann ansonsten bestehen, falls der Moderator diesen Schritt zu knapp oder zu schnell durchführt. Wenn kein Ausgangssystem existiert, dann gibt es hoffentlich auch keine oder weniger unnötige implizite Annahmen.

Am besten sind für diesen Ansatz vermutlich Entwicklungswerkzeuge geeignet, mit denen schnell funktionstüchtige und gleichzeitig überschaubare Lösungen entwickelt werden können. Mit einem leichtgewichtigen Werkzeug oder einer Lernumgebung wie BlueJ[1] lässt sich ein solches Szenario evtl. besser umsetzen als mit einer eher schwerfälligen Entwicklungsumgebung wie Eclipse[2].

Skizze eines Teachlets ohne Ausgangssystems in BlueJ

Es folgt ein grober Entwurf eines Teachlets zum Entwurfsmuster *Beobachter*, das unter Verwendung von BlueJ ohne Ausgangssystem auskommt. Die Zielgruppe sind Teilnehmer, die mit Java vertraut sind und neben sonstigen Grundlagen mindestens das *Java Collection Framework* in seinen Grundzügen kennen, da `Set` und `HashSet` verwendet werden. Dieser Entwurf muss natürlich noch weiter und detaillierter ausgearbeitet werden, bevor er als Teachlet veranstaltet werden kann.

[1] Lernumgebung für Java, die das Visualisieren und interaktive Erstellen von Exemplaren von Klassen ermöglicht. http://www.bluej.org/

[2] Quelloffene Entwicklungsumgebung für diverse Programmiersprachen einschließlich Java. http://www.eclipse.org/

Das Szenario ist, dass die UEFA die Kameraführung beim Filmen von Fußballspielen automatisieren möchte. Dazu wird der Ball mit einem Mikrochip und Sensoren ausgestattet, mit denen er jederzeit seine eigene Position und Geschwindigkeit auslesen kann. Jede der vielen Kameras, die um das Spielfeld verteilt sind, soll die Möglichkeit bekommen, sich an den Ball zu „heften“ und über all seine Bewegungen informiert zu werden, bis sie sich wieder aus diesem Nachrichtenkanal ausklinkt. Natürlich soll das System in Zukunft auch auf andere Dinge als Kameras erweiterbar sein.

Der Moderator kann dann mit den Teilnehmern diskutieren, wie sich diese Aufgabe prinzipiell lösen lässt. Denkbar wäre z.B., dass jede Kamera einen Timer bekommt und alle paar Millisekunden den Ball sucht (diese Variante ist wegen der Auslastung der Prozessoren nicht praktikabel). Womöglich kommen die Teilnehmer von selbst darauf, dass der Ball eine Liste seiner Beobachter halten und bei Veränderungen seines Zustands jeden einzeln informieren kann. An der Schnittstelle des Balls muss es dann die Möglichkeit geben, sich als Beobachter ein- und auszutragen. Damit ist das Beobachtermuster auch bereits erschlossen. Der Moderator kann es dann noch einmal im Ganzen vorstellen.

Es kann ein Klassendiagramm erstellt werden, bevor der Entwurf in BlueJ in Code umgesetzt wird. Aufgrund der besonderen Fähigkeiten von BlueJ kann auf eine interaktive Oberfläche komplett verzichtet werden – es reicht, wenn die Kameras bei jedem Ereignis eine Zeile auf der Konsole ausgeben. Ein übergeordnetes Modell für das Spiel oder das Stadion ist überhaupt nicht nötig, da Exemplare der Klassen `Ball` und `Kamera` in BlueJ interaktiv erzeugt und manipuliert werden können.

Ausgehend von einem leeren BlueJ-Projekt reicht es also, zwei Klassen und ein Interface zu erzeugen: `Ball`, `Kamera` und das Interface `BallBeobachter`. Letzteres braucht lediglich eine Operation mit der folgenden Signatur:

```
void ballEreignis(int x, int y, int geschwindigkeit)
```

Die Klasse `Kamera` implementiert dann das Interface, speichert eine im Konstruktor übergebene ID (für die Ausgabe) und hält eine Referenz auf einen Ball, welche zunächst `null` ist. Ansonsten hat sie eine Methode `beobachteBall(Ball ball)`, die sie ggf. vom letzten Ball abmeldet, den Ball speichert und sich dort als Beobachter anmeldet. Natürlich muss die Kamera die durch das Interface vorgegebene Operation implementieren. In dieser Methode können die Daten in einem hübsch formatierten String auf der Konsole ausgegeben werden.

Der Ball enthält den für das Beobachtermuster typischen Code zur Verwaltung von Beobachtern, hier mit einem `Set<BallBeobachter>` umgesetzt (im Konstruktor wird ein entsprechendes `HashSet` erzeugt). Außerdem gibt es eine `ballGetreten`-Operation, in der die Position und Geschwindigkeit einfach auf zufällige Werte gesetzt und dann an alle Beobachter übermittelt werden.

Die drei Java-Dateien sind im Anhang abgedruckt (ab Seite 57). Aus Gründen der Übersichtlichkeit und Geschwindigkeit beim Schreiben im Plenum wurde auf Kommentare im Code komplett verzichtet. Der Moderator möge im Vorfeld abwägen, ob die Kommentierung des Codes während der Erstellung didaktisch sinnvoll genug ist, um die dadurch verlorene Zeit in der Choreographie wert zu sein.

Wenn die zwei Klassen und das Interface geschrieben und kompiliert sind, dann können in BlueJ interaktiv ein paar Kameras und ein Ball erzeugt werden. Die Kameras können

dann am Ball angemeldet werden und der Ball kann ein paar mal „getreten“ werden – im Terminalfenster wird dann immer die Ausgabe der aktuell als Beobachter angemeldeten Kameras erscheinen.

In dem von BlueJ automatisch erzeugten Klassendiagramm lässt sich sehr gut zeigen, dass das Beobachtermuster zu einem Entwurf ohne zyklische Abhängigkeiten führt.

5.3.4. Verwendung alternativer Eingabewerkzeuge

In bisherigen typischen Teachlets werden zwei weitgehend unabhängige visuelle Medien eingesetzt: der Beamer mit den Präsentationsfolien oder der Software, die vom Moderator per Maus und Tastatur gesteuert werden, sowie eine Kreidetafel bzw. ein Whiteboard, an der bzw. dem Diagramme gezeichnet werden.

In einem Teachlet der Teachlet-Werkstatt im Jahr 2010 wurden diese Medien durch die Verwendung eines SmartBoards^{TM} stärker miteinander verbunden. In dem Fall wurden die Interaktionsmöglichkeiten des SmartBoards^{TM} sowohl als Aufwertung der Code-Inspektion als auch als Hilfsmittel bei der Erstellung von Klassen- und Objektdiagrammen verwendet und wurden von den Teilnehmern des Teachlets allgemein als positiv bewertet[FS10].

5.4. Jenseits von Teachlets

Im Abschnitt 5.2 wurden viele der identifizierten zentralen Aspekte als definierende und unverzichtbare Merkmale von Teachlets herausgestellt. Im Einzelnen sind das: *Moderator*, *Teilnehmer*, *Interaktion*, *Aufgabe*, *Lernziel*, *Lösungsraum*, *Architektur*, *Rechner* sowie *Gemeinsame Sicht*. Wenn mindestens einer dieser neun Aspekte nicht gegeben ist, dann bewegen wir uns also außerhalb des durch das Teachlet-Konzept gesetzten Rahmens. Das bedeutet natürlich nicht, dass Überlegungen in diese Richtung keinen Wert hätten. In diesem Sinne sollen abschließend ein paar Gedankenanstöße gegeben werden, wie einige dieser Teachlet-ähnlichen Methoden aussehen könnten.

Ohne Moderator:
Irgendwie müssen die Teilnehmer zur Aufgabe finden. Wenn dies nicht durch einen Moderator (oder eine ähnliche Rolle, etwa einen Betreuer) geschieht, dann ist die Aufgabe evtl. schriftlich gegeben und es existiert eine Sammlung von Unterlagen zu einem bestimmten Thema. Dann könnten Teilnehmer alleine oder in der Gruppe sich anhand der Unterlagen eine Aufgabe erschließen, sich z.B. in ein Entwurfsmuster einlesen und dann eine Lösung zum gestellten Problem ohne Anleitung entwerfen. Das Prinzip wäre das einer Materialsammlung zum Selbststudium von Softwaretechnik-Inhalten anhand konkreter Probleme. Da die Rolle des Moderators in anderen Varianten der Methode eine Person mit viel Erfahrung mit dem Lerngegenstand erfordert und bindet, ist in dieser Variante der ökonomische Verlust im Falle des Misslingens ggf. weniger schwerwiegend. Wie bei anderen Methoden des Selbststudiums ist allerdings auch eine gesteigerte Eigenmotivation der Teilnehmer nötig.

Ohne Teilnehmer:
Wenn der Begriff des Teilnehmers so verstanden wird, dass jeder ein Teilnehmer ist,

der als Konsument von der Methode profitieren soll, dann ist eine Vorgehensweise ohne Teilnehmer offensichtlich sinnlos. Wenn Teilnehmer und Moderator als voneinander abgegrenzte Rollen zu verstehen sind, kann die im obigen Absatz beschriebene Idee so aufgefasst werden, dass sie die Trennung von Teilnehmern und Moderatoren aufhebt, so dass alle gemeinsam auf dem gleichen Level arbeiten. Hier ließen sich Parallelen zu Methoden wie *Expertengruppen* ziehen.

Ohne Interaktion:
Wenn zwischen Teilnehmern und Moderator keine Interaktion stattfindet, dann bewegen wir uns wieder im Bereich des traditionellen Vortrags, in dem Wissen ganz konventionell von einer Person zu vielen anderen fließt. Diese Perspektive bietet aus methodischer Sicht nicht viel Neues.

Ohne Aufgabe:
Wenn es keine Aufgabenstellung gibt, dann drängt sich die Frage auf, woran und wie gearbeitet werden soll. Angenommen, die Teilnehmer sollen aktiv sein und nicht nur zuhören. Vielleicht wäre es dann möglich, ohne eine vorbereitete Aufgabe heranzugehen und die Teilnehmer zu fragen, welches Problem sie beschäftigt oder welche Frage sie bisher nicht klären konnten (bezogen auf ein Lernziel oder auch völlig frei innerhalb eines bestimmten Themengebiets). Das scheint eine sehr offene Methode zu sein, wäre aber in der Praxis an das Problem gebunden, dass Aufgaben und Fragen des richtigen Kalibers spontan von den Teilnehmern kommen müssten. Eine Methode völlig ohne jegliche Art von Aufgaben erscheint nicht zweckmäßig.

Ohne Lernziel:
Im Gegensatz zum vorigen Absatz sind hier schnell diverse Möglichkeiten zu erkennen: Ein Moderator könnte bspw. ohne feste Zielvorstellung ein interessantes Ausgangssystem vorstellen und den Teilnehmern Aufgaben stellen (oder sie selbst Problemstellen finden lassen) und dann wie bei einem Teachlet gemeinsam Lösungen erarbeiten oder auch die Teilnehmer einzeln (oder im Paar) an eigenen Rechnern arbeiten lassen und hinterher Ergebnisse zusammenstellen. Ein solches Vorgehen wäre weniger zielorientiert als ein Teachlet und dafür eher explorativ. Zum Kennenlernen einer bestehenden Codebasis könnten diese Ideen fruchtbar sein.

Ohne Lösungsraum:
Wenn es keinen Lösungsraum im Teachlet-Sinn gibt, dann heißt das, dass es für die Aufgabe evtl. nur eine einzige Lösung gibt (oder dass nur eine einzige Lösung zugelassen wird). Methodisch können solche Vorgehensweisen bspw. sinnvoll sein, wenn die Teilnehmer mit der Sprache, in der entwickelt wird, nicht ausreichend vertraut sind um sich einen Lösungsraum erschließen zu können, oder wenn eine ausführliche Diskussion über Lösungsvarianten aus anderen Gründen nicht in Frage kommt. In diesem Sinne ist gegen den Einsatz eines solchen Konzepts nichts einzuwenden.

Ohne Architektur:
Wenn keine (Software-)Architektur eine Rolle spielt (weder im großen noch im kleinen Maßstab), dann hat das ohne Zweifel auch Auswirkungen auf den Lösungsraum – seine

typische Größe erreicht er bei Teachlets dadurch, dass ein Architekturproblem gestellt wird. Von diesem Zusammenhang abgesehen lassen sich natürlich auch alle möglichen anderen Arten von Problemen im Plenum systematisch untersuchen.

Ohne Rechner:
Wenn kein Computer verwendet wird, fällt die Arbeit an laufender bzw. direkt lauffähiger Software notwendigerweise weg. Natürlich kann man trotzdem an der Tafel Klassendiagramme zeichnen, über Entwürfe sprechen und noch vieles mehr tun, allerdings ist dieses Vorgehen ohne echtes „Anschauungsmaterial“ für die Teilnehmer viel weniger mitreißend. Eine praktische Vermittlung von Informatik-Inhalten ohne Verwendung eines Rechners erscheint nicht zweckmäßig.

Ohne gemeinsame Sicht:
Die Betrachtung des Lernziels und der Software im Plenum gehört zu einem Teachlet ganz klar dazu. Wenn nicht mit einer gemeinsamen Sicht gearbeitet wird, sondern die Teilnehmer etwa an Einzelrechnern sitzen, dann entsteht keine mit Teachlets vergleichbare Diskussionskultur. Damit läge man eher im Bereich der traditionellen Rechnerübungen, in denen eine Programmieraufgabe gestellt und von den Teilnehmern einzeln (oder im Paar) bearbeitet wird.

6. Fazit

In dieser Arbeit wurde das Teachlet-Konzept dargestellt und in den Kontext der Pädagogik eingeordnet, Praxisberichte evaluiert, die Teachlet-Definition kritisiert und überarbeitet, das Konzept weiter eingegrenzt und einige neue Impulse für zukünftige Teachlets geliefert. Hoffentlich kann die Frage, was denn eigentlich genau ein Teachlet sei und was es bringe, mit der Lektüre dieser Arbeit geklärt werden. Die überarbeitete Definition und die Klärung der wichtigsten Begriffe trägt hoffentlich dazu bei, dass der Diskurs zum Thema auf einer sicheren Grundlage weitergeführt werden kann.

6.1. Ausblick

Die Veranstalter der Teachlet-Werkstatt verfügen zum Zeitpunkt der Fertigstellung dieser Arbeit über eine Sammlung von 41 fertigen und dokumentierten Teachlets. Diese sind mit Thema, Versionsnummer und Autor(en) in einem Archiv hinterlegt, das jedoch keine strukturierte Suche oder Erfassung der nötigen Metadaten ermöglicht. Um dem wachsenden Bestand gerecht zu werden, böte es sich an, eine bessere Möglichkeit der Archivierung zu entwickeln. In diesem Zuge könnten die vorhandenen Teachlets auch gleich strukturiert abgelegt und statistisch erfassbar gemacht werden. Zu dem Thema befindet sich am Arbeitsbereich Softwaretechnik eine Bachelorarbeit in der Vorbereitungsphase, deren Ergebnis noch nicht abzusehen ist.

Immer wieder hat sich auch die Frage gestellt, inwieweit sich das Prinzip der Teachlets auf andere Bereiche als die Softwareentwicklung anwenden lässt. Für diese Arbeit ist sie bewusst ausgeklammert, für Forschung mit stärkerem Bezug zur Pädagogik wäre es ein lohnenswertes Thema.

Nicht zuletzt bleibt zu hoffen, dass Teachlets auch weiterhin eingesetzt und weiterentwickelt werden. Die in dieser Arbeit genannten neuen Einsatzkontexte und Variationsmöglichkeiten sind mit einiger Sicherheit nicht erschöpfend. Eine weitergehende Verwendung und Erforschung des Konzepts wäre wünschenswert.

Anhang

Interview-Protokolle

Interview mit Axel Schmolitzky

Dieses Interview wurde am 7. September 2010 in Raum D-206 am Informatikum der Universität Hamburg durchgeführt. Es begann um 14 Uhr und dauerte ca. 35 Minuten.

Julian Fietkau:
Okay, vielleicht magst du dich noch mal kurz vorstellen, fürs Band?

Axel Schmolitzky:
Ja, mein Name ist Axel Schmolitzky, ich bin akademischer Rat im Arbeitsbereich Softwaretechnik hier an der Uni Hamburg in der Informatik. Ich bin seit 2001 hier im Arbeitsbereich tätig. Vorher war ich noch wissenschaftlicher Assistent[1], ich bin als Post-Doc nach Hamburg gekommen und habe dann unter Anderem auch in der Lehre etliche Dinge vertreten, die hier noch neu waren: Dinge wie Objects First[2], BlueJ[3], Vererbungskonzepte konsequent untersuchen und Ähnliches. So viel, glaube ich, erst mal zu meiner Person.

Julian Fietkau:
Hier soll es ja heute erst mal um Teachlets gehen. Nun ist gerade auch die Verbindung zwischen dir und Teachlets ja eine ziemlich große und prägnante. Da würde mich erst mal allgemein interessieren, was du bisher so mit Teachlets zu tun hattest, was du damit für Erfahrungen gemacht hast.

Axel Schmolitzky:
Es ist natürlich, sagen wir, quasi müßig die Frage zu stellen, weil ich sozusagen derjenige bin, der die Teachlets in die Welt gebracht hat. *(lacht)* Die sind entstanden im Rahmen eines kleinen Notfalles, weil ich in einem Seminar - damals haben wir noch regelmäßig ein Rahmenwerksseminar durchgeführt, ich glaube jedes Semester sogar. In diesem Rahmenwerksseminar ging es natürlich um Rahmenwerke, also Frameworks, objektorientierte Rahmenwerke. Da ist ein Vortragstermin ausgefallen und ich war da sehr pflichtbewusst und dachte, ich müsste mir da irgendwas einfallen lassen, wie ich diesen Termin befüllen kann. Ich habe das in der SWT-Gruppenbesprechung mal kurz angesprochen und Heinz Züllighoven[4] sagte dann: „Mach doch vielleicht eine kleine Programmierwerkstatt, in der du ein Entwurfsmuster mit den Leuten durchsprichst, also wie das implementiert wird.“ Die Idee schien mir ganz attraktiv und ich habe dann tatsächlich aus Schulungsfolien der WPS (das ist eine Firma, in der Heinz Züllighoven Geschäftsführer ist) etwas zum Entwurfsmuster *Chain of Responsibility*, glaube ich, gemacht. Und das habe ich dann

[1] Gemeint ist: vor der Zeit als akademischer Rat. Axel Schmolitzky ist 2001 an die Universität Hamburg gekommen, war dann zunächst wissenschaftlicher Assistent und später akademischer Rat.
[2] Lehransatz, nach dem den Lernenden die Grundkonzepte der Objektorientierung (Klassen und Exemplare) als erstes, also noch vor imperativen Kontrollstrukturen, vermittelt werden.
[3] Lernumgebung für Java, die das Visualisieren und interaktive Erstellen von Exemplaren von Klassen ermöglicht.
[4] Prof. Dr.-Ing. Heinz Züllighoven, Professor für Softwaretechnik an der Universität Hamburg.

in diesem allerersten Teachlet umgesetzt, habe dann tatsächlich ein Ausgangssystem gezeigt, den Studierenden eine Aufgabe gestellt und gesagt: Wie können wir das jetzt hier lösen? Das war dann die Geburtsstunde der Teachlet-Idee.

Ich habe das dann in einem Projektseminar ausführlicher untersucht, in dem ich da mit fünfzehn, sechzehn Leuten in einer 4-SWS-Veranstaltung das Teachlet-Konzept, das ich damals schon ein bisschen stärker formalisiert hatte, und noch einige andere Ideen eingebracht habe. Seitdem veranstalte ich jedes Jahr ein mal eine sogenannte *Teachlet-Werkstatt*, nämlich ein Seminar, in dem die Studierenden selbst Teachlets entwerfen und an den Teilnehmern der Werkstatt ausprobieren.

Julian Fietkau:
Also du hattest durchgehend mit Teachlets immer wieder zu tun, kann man sagen.

Axel Schmolitzky:
Das ist richtig. Die begleiten mich eigentlich seit... Dieses Seminar, dieses Projektseminar, dieses 4-SWS-Seminar, das war 2003 soweit ich mich erinnere... 2004 war das. Seit 2004 mache ich eigentlich jedes Jahr eine Teachlet-Werkstatt. Von daher bin ich tatsächlich sehr stark mit Teachlets in Kontakt.

Julian Fietkau:
Und wenn du die in Kontrast setzt zu anderen Lehrformen wie Vorlesungen und sowas, damit hast du ja auch Erfahrungen, wo sind da so die wichtigen Unterschiede? Wie zeichnen sich Teachlets da aus?

Axel Schmolitzky:
Ich denke, die große Stärke von Teachlets liegt tatsächlich in der hohen Interaktivität, die quasi „Programm“ ist in diesem Ansatz, also es geht gar nicht ohne Interaktivität. Das ist natürlich ein großer Unterschied zu klassischen Vorlesungen, die meist sehr in eine Richtung gehen: Der Dozent redet und alle anderen hören zu. Auch deutlich interaktiver als zum Beispiel ein Seminar, wo üblicherweise auch eine Person einen Vortrag hält, wo dann am Ende wenn man Glück hat noch eine Diskussion aufkommt zu dem Thema. Aber bei den Teachlets ist es so, dass zwar eine Choreographie da ist, aber dann doch die Teilnehmer relativ stark mitbestimmen können, was denn da passiert. Ich denke, dass die Stärke gerade auch in dem Bereich liegt, wo es darum geht, einen Entwurf zu diskutieren. Ich glaube das ist über die Jahre immer deutlicher geworden: Dass das Teachlet-Konzept sich gut eignet für Bereiche, in denen es einen gewissen Lösungsraum gibt oder einen gewissen Entwurfsraum. Nicht einfach nur eine typische, offensichtliche Lösung, die sollen alle kennenlernen und die programmiert man dann herunter. Das wäre auch vorstellbar, aber das würde ich dann nicht mehr Teachlet nennen. Ich glaube das, was mir schon sehr wichtig ist, ist, dass ein Entwurfsraum besteht, der am besten schön aufgespannt wird durch den Moderator, der sozusagen die Szenerie setzt. Und dann sollen sich in diesem Raum, der da definiert wird, alle Teilnehmer irgendwie austoben mit Vorschlägen und mit Diskussionen über verschiedene Entwürfe und Entwurfsalternativen. Das ist die Grundidee der Teachlets.

Julian Fietkau:
Die Teachlets haben sich bisher soweit ich weiß schwerpunktmäßig mit Entwurfsmustern befasst, objektorientierten Entwurfsmustern, richtig?

Axel Schmolitzky:
Das ist richtig, das war deutlich der Schwerpunkt über die letzten Jahre. Wir haben inzwischen zu fast allen Entwurfsmustern aus dem Gamma et al.[5] ein Teachlet, oder mehrere Teachlets zu einigen Entwurfsmustern. Wir haben glaube ich nur zum Singleton und zur Template Method kein Teachlet. Ansonsten haben wir eigentlich zu allen Entwurfsmustern mal etwas gehabt. Das ist aber nicht der einzige Bereich, in dem sie gut wirken können. Wir haben unter anderem auch, weil das Seminar auch eigentlich so heißt – es heißt nämlich formal nicht Teachlet-Werkstatt sondern „Konzepte objektorientierter Programmiersprachen“ – da geht es auch um Programmiersprachkonzepte. Wir haben auch versucht, Programmiersprachkonzepte als Lernziele in Teachlets zu installieren, das heißt zum Beispiel zu sagen: multiple Implementationsvererbung in C++. Da war dann die Idee, dass man auch ein Setting herstellt, dass man ein Ausgangssystem zeigt, und dann sagt: Das und das soll jetzt passieren, wie lässt sich das am besten lösen? Dann sollte es so sein, dass die multiple Implementationsvererbung die Lösung ist, die sich dafür anbietet, und dass man auf die Weise ein Programmiersprachenkonzept mal als Lösung für ein Entwurfsproblem sehen kann. Das hat mal weniger gut und mal besser geklappt. Wir haben zum Beispiel mehrfach versucht, das Konzept von Closures in einem Teachlet darzustellen. Aber wir habe nauch so etwas gehabt wie Multiple Dispatch in Programmiersprachen, dass also nicht nur nach dem Typ des Empfängers dynamisch gebunden wird, sondern auch nach den Typen mehrerer Parameter eines Methodenaufrufs, und ähnliche Dinge.

Das, würde ich sagen, war anfangs gleichberechtigt, ist inzwischen dann doch in den Hintergrund getreten. Das liegt ein bisschen daran, dass die Veranstaltung von einer Hauptstudiums- zu einer Bachelorveranstaltung geworden ist, für Bachelorstudierende, die oft erst im vierten oder sechsten Semester sind, während früher die Teilnehmer üblicherweise mindestens im sechsten, eher im achten, zehnten oder zwölften Semester des Informatikstudiums waren und dann einfach schon mehr Hintergrund hatten was die Programmiererfahrung angeht. Dadurch sind wir jetzt bei den Inhalten sehr stark eher bei den Entwurfsmustern, um die Entwurfsmuster besser kennen zu lernen, und weniger darin, zu sagen: Hier ist eine neue Programmiersprache, in der soll jetzt das und das Konzept mal kennengelernt werden. Da musste man mindestens erst mal eine halbe Stunde bis eine Stunde investieren um die Sprachmechanismen der Sprache vorzustellen. Das war zum einen vom zeitlichen Aufwand nicht so schön, zum anderen ist es relativ anspruchsvoll, eine andere Programmiersprache mal eben herzunehmen und dann in der Runde vorzustellen. Das scheint mir für Bachelorstudierende in einem relativ frühen Stadium des Studiums eher eine Überforderung zu sein. Das ist so mein Eindruck, dass wir deswegen in letzter Zeit sehr stark zu den Entwurfsmustern zurückgekehrt sind.

Dann gibt es noch eine dritte Anwendungsmöglichkeit, die wir bisher nur einmal pro-

[5] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston : Addison-Wesley, 1995

biert haben, nämlich Algorithmen. Auch Algorithmen können mit einem Teachlet vermittelt werden. Da ist aber dann auch eher die Idee, nicht den Algorithmus selbst zu vermitteln. Wir haben festgestellt, eigentlich muss der Algorithmus von seinem Grundkonzept bei allen verstanden sein bzw. der Moderator sollte den Algorithmus vorstellen. Spannender ist dann die Frage: Wie kann man diesen Algorithmus implementieren? Weil das oft von der jeweiligen Programmiersprache abhängig ist, wie gut oder wie schlecht das geht. Da ist die Idee gewesen, dass man, selbst wenn man in einer Programmiersprache ist, immer noch einen Entwurfsraum hat, wie man denn diesen Algorithmus umsetzt.

Wir hatten das Beispiel mal mit dem Kürzeste-Wege-Algorithmus[6] von Dijkstra[7]. Der kann auf ganz unterschiedliche Art und Weise implementiert werden. Da wäre es sehr spannend gewesen, wenn man tatsächlich mal geguckt hätte, welche Möglichkeiten es da so gibt. Das hat nicht so gut geklappt, weil auch da die Voraussetzung ist, dass erst mal der Moderator den Algorithmus sehr, sehr gut kennen muss, und auch die verschiedenen Implementierungsmöglichkeiten, zum anderen aber auch die Teilnehmer den Algorithmus gut kennen sollten, das heißt also, der Moderator gut darin sein muss, das am Anfang darzustellen, so dass alle auf dem Stand sind: Ja, wir haben den Algorithmus verstanden, jetzt müssen wir uns nur noch überlegen, wie wir das implementieren. Ich glaube die Idee ist eigentlich gut, aber auch möglicherweise etwas zu anspruchsvoll.

Julian Fietkau:
Das waren jetzt also mögliche Themen. Du hattest eben schon die Interaktivität als eine wichtige Eigenschaft von Teachlets genannt. Was siehst du sonst noch für Vorteile und Möglichkeiten des Konzepts? Was sind die Stärken von Teachlets?

Axel Schmolitzky:
Wenn ich auf Entwurfsmuster sehe, dann sehe ich, dass die ein Problem haben, das eigentlich ihre Stärke ist. Entwurfsmuster sind Abstraktionen von verschiedenen Entwurfsmöglichkeiten. Das heißt, ein Entwurfsmuster abstrahiert und fasst zusammen, kondensiert bestimmte Dinge, Ideen, Ansätze, Konzepte, und fasst das sehr knapp zusammen. Das ist auf der einen Seite sehr praktisch für Leute, die viel Programmiererfahrung haben, die sehr viel mit Entwürfen umgehen, ein knappes Vokabular zu bekommen, also Dinge zu benennen, die ihnen ständig über den Weg gelaufen sind. Aber es ist relativ schwierig für Studierende, die diese Entwurfsmuster gerade erst erlernen sollen, oder überhaupt Entwurfsprobleme erlernen sollen, aus diesen abstrakten Beschreibungen etwas abzuleiten. Ich weiß, dass man den Gamma so durchlesen kann und eigentlich hinterher nicht viel schlauer ist als vorher. Das geht sehr gut, die Gefahr ist sehr groß. Da sehe ich dann die große Stärke, dass bei einem Teachlet immer an einem konkreten Software-System, das einigermaßen verständlich ist für jemanden, der Java in den Grundzügen gut verstanden hat, sehr stark konkretisiert wird. Dass man wirklich sieht: Ah ja, hier ist jetzt das Muster in dieser Form umgesetzt. Ich glaube das ist etwas, was eine große Stärke von

[6] Auch: Dijkstra-Algorithmus, vgl. [CLR04]
[7] Edsger W. Dijkstra, niederländischer Informatiker, bekannt für den Dijkstra-Algorithmus, die Erfindung von Semaphoren und mehr.

Teachlets ist, dass sie etwas relativ Abstraktes sehr konkret verdeutlichen können, und dass sie dann aber auch die Chance bieten, dass man das noch mal reflektiert und dann sozusagen vom Allgemeinen zum Speziellen über das Verständnis des Speziellen den Teilnehmern die Möglichkeit gibt, noch mal die Abstraktion auch besser zu verstehen. Dass man das auch auf andere Fälle transponiert, weil man den einen gut verstanden hat. Ich glaube das ist eine weitere Stärke.

Jetzt muss ich überlegen, gibt es noch eine weitere Stärke...? Also die Interaktivität, die Konkretisierung von abstrakten Konzepten um sie besser verstehen zu können... Ein Seiteneffekt, den ich auch recht positiv finde, ist dieses gemeinsame Programmieren, üblicherweise mit einer Entwicklungsumgebung, die auch recht anspruchsvoll ist – meistens ist es Eclipse – dass tatsächlich über diese Situation, die sich da ergibt, alle Teilnehmer immer noch ein bisschen was lernen können über die Entwicklungsumgebung. So etwas Banales wie: Mit der Tastenkombination kannst du das an der Stelle viel schneller machen. Das ist so Handwerkszeug, konkretes Wissen, das auf diese Weise sehr angenehm weiter getragen werden kann. Das ist ein Randeffekt, den habe ich auch beobachtet und den finde ich auch sehr positiv. Es geht bei einem Teachlet schon sehr stark um Software und um Softwareentwicklung. Das steht schon sehr im Vordergrund.

Julian Fietkau:
Siehst du an dem Konzept auch irgendwo harte Grenzen? Gibt es Kontexte wo du sagen würdest: Da können Teachlets auf keinen Fall eingesetzt werden, oder nur eingeschränkt?

Axel Schmolitzky:
Ich habe schon unterschiedliche Versuche unternommen, Teachlets in andere Bereiche einzubringen, oder die Erfahrungen mit Teachlets. Ich halte meine Vorlesungen im Allgemeinen auch so, dass ich eigentlich immer mindestens ein mal in der Vorlesung meine Entwicklungsumgebung, die ich benutze, entweder BlueJ oder auch Eclipse, starte und einfach live programmiere in der Veranstaltung. Ich halte das für sehr nützlich. Das ist sicher auch ein bisschen motiviert über die Erfahrung mit Teachlets. Aber ich habe es selten in meinen Vorlesungen geschafft, eine Teachlet-artige Situation herzustellen, nämlich ein Ausgangssystem zu motivieren und eine Aufgabe zu stellen, die dann zu einer Entwurfsdiskussion führt. Das ist einfach in dem Rahmen von mehreren hundert Zuhörern schlecht zu machen, ist mein Eindruck. Es ist tatsächlich auch so, dass eine Veranstaltung wie eine Vorlesung sehr viel mehr Druck hat, einen bestimmten Stoff durchzubringen. Das ist mit den Teachlets nicht so gut möglich. Ich glaube, dass es da tatsächlich eher darum geht, etwas zu üben, nämlich die Fähigkeit, Entwürfe zu diskustieren, zur Diskussion zu stellen, aber auch Lösungen mit anderen Vorschlägen zu vergleichen. Das braucht Zeit, das ist auch eine notwendige Fähigkeit beim Programmieren, die aber in eine Vorlesung nicht gut hereinpasst. Das ist sicherlich eine Grenze.

Die Teilnehmerzahl muss aber nicht unbedingt eine Grenze sein, denn wir haben ja auch tatsächlich schon Erfahrungen gesammelt, jetzt gerade kürzlich mit einem Teachlet, in Anführungsstrichen, beim iPhone-PowerDay, der hier in Hamburg stattgefunden hat. Da waren es, korrigier mich, auch mehrere hundert Teilnehmer? *(JF: 150 oder so.)* Also 150 Teilnehmer, ein relativ großer Rahmen, ähnlich wie eine Vorlesung. Da ist auch

explizit für eine Stunde ein Teachlet eingeplant worden, das auch tatsächlich, so wie mir gesagt wurde, relativ gut funktioniert hat. Das heißt, der Moderator hat versucht, aus dem Publikum Zurufe zu bekommen, die ihn weiterführen bei dem, was er da gerade live programmiert. Und das hat funktioniert, es kamen tatsächlich die entsprechenden Bemerkungen aus dem Publikum. Das heißt da ist eine Interaktivität möglich.

Natürlich ist es dann nicht möglich, alle gleichermaßen zu beteiligen. Aber wenn ich ehrlich bin, ist es auch bei den Teachlets, die wir hier im kleinen Rahmen in der Teachlet-Werkstatt hatten, selten so gewesen, dass alle beteiligt waren. Es sitzen immer zwei, drei, vier, fünf Leute nur da und hören sich das an, was da an Entwurfsdiskussionen passiert und beteiligen sich an den Diskussionen nicht. Von daher ist das nicht wirklich etwas, was in der großen Runde anders ist. Man hat immer eine handvoll von Leuten mindestens, die sich aktiv beteiligen. Ob das jetzt fünf von 500 oder fünf von zehn sind, ist eigentlich fast egal, solange tatsächlich mehrere Leute ihre Ideen einbringen und diese Ideen zur Diskussion gestellt und gegenübergestellt werden. Ich glaube, das ist eine große Stärke, dass man nicht nur einen Blickwinkel einer Person darstellt, sondern dass mindestens die von fünf bis zehn Personen dargestellt werden. Das ist, glaube ich, fast immer vorteilhaft für das Verständnis. Auch für alle, die sich nur passiv beteiligen.

Julian Fietkau:
Könntest du dir auch so etwas vorstellen wie, ganz abenteuerlich gedacht, einfach mal Teachlets ohne physische Anwesenheit? Zum Beispiel per Teleconferencing, Videochat oder so etwas? Das wäre ja auch noch mal ein völlig anderer Kontext.

Axel Schmolitzky:
Klar, könnte ich mir vorstellen. Ich persönlich bin ein großer Befürworter von Face-to-Face-Kommunikation, also direktes Gespräch und direkte Diskussion. Das ist auch eine der Praktiken aus dem XP, dem *Extreme Programming*[8], durch das ich sehr stark geprägt bin die letzten Jahre. Dieses *sit together*, das hat natürlich schon etwas. Man soll in einem Raum sitzen, man soll zusammenarbeiten als Team, möglichst dicht beieinander. Das ist schon etwas, was meiner Meinung nach die Effektivität erhöht. Aber klar könnte ich mir das auch vorstellen. Das ist dann aber auch eine Frage vom Flüssigsein mit dem Umgang mit diesen ganzen Medien. Es gibt Leute, die machen solche Online-Vorlesungen und auch Online-Vorlesungen mit der Möglichkeit, dass die Teilnehmer Einfluss nehmen oder Fragen stellen können. Wenn man die Software, die da zum Einsatz kommt, da gibt es so verschiedene Systeme, wenn man die gut beherrscht und im Umgang wie gesagt flüssig ist, dann kann ich mir schon vorstellen dass man dann auch so eine Entwurfsdiskussion mal führen könnte. Denkbar wäre das. Ich meine, den Bildschirm, den man mit dem Beamer an die Wand wirft damit alle sehen was da passiert, den kann man genau so auch im Internet an alle Teilnehmer raussenden und alle können sehen was passiert und alle können dann tatsächlich Einfluss darauf nehmen. Das ist durchaus denkbar. Von meiner persönlichen Eignung her würde ich eher sagen, ich bin nicht der Richtige dafür. Aber das Konzept spricht, glaube ich, nicht dagegen.

[8] Eine Methodik für agile Softwareentwicklung, vgl. [Bec99]

Julian Fietkau:
Dann jetzt noch so die Frage des Tages, die mich ja noch interessieren würde. Ohne irgendwelche Namen von Moderatoren zu nennen: Hast du es schon erlebt, dass Teachlets so richtig schiefgegangen sind?

Axel Schmolitzky:
Puh, so richtig schiefgegangen... Es gab einen Fall in der allerersten Teachlet-Werkstatt, aber das war auch wirklich kein... normaler Fall. Das war eine Person, die auch in anderen Kontexten schon auffällig war und überhaupt nicht in der Lage war, die Grundidee des Entwurfsmusters zu erfassen, auch nicht in der Lage war, systematisch ein Softwaresystem weiterzuentwickeln, und ich es tatsächlich nicht geschafft habe, die ganze Zeit dabei zu bleiben, weil ich mich sonst zu sehr aufgeregt hätte. Ich hab also von vornherein gewusst, dass es in die Hose geht. Es ging in die Hose, es war eine Zumutung für alle Beteiligten und ich musste nicht viel dazu sagen. Ich bin, wie gesagt, zwischendurch rausgegangen, sonst wäre ich geplatzt. Als ich wieder reinkam, war dann das Peer-Feedback so ausgeprägt, dass auch alle anderen sehr klar dieses ausgesagt haben. Das war das einzige Mal, aber das war eigentlich kein realer, echter Fall. Ansonsten richtig schiefgegangen, da muss ich mal überlegen... Nein, so richtig schiefgegangen... Es kommt immer irgendetwas dabei herüber. Selbst, wenn man nur die Folien zeigt zu dem Entwurfsmuster und dann versucht, das ein bisschen auf das konkrete System anzuwenden, selbst wenn da keine große Entwurfsdiskussion aufkommt, ist es trotzdem immer so, dass genügend Inhalt ankommt, würde ich jetzt mal sagen. Wie gesagt, abgesehen von diesem allerersten Fall gab es nie einen Totalausfall. Nicht, dass ich mich erinnere.

Julian Fietkau:
Das ist doch mal ein Werbespruch: Das Teachlet, die Methode, die sogar klappt, wenn sie schiefgeht.

Axel Schmolitzky:
(lacht) Ja, das liegt natürlich auch einfach daran, dass es eben nicht nur auf den Moderator ankommt, sondern auch auf das Team. Ich habe es tatsächlich schon erlebt, dass ein Teachlet sehr stark getragen wurde durch sehr gute Teilnehmer, obwohl der Moderator nicht sehr gut war. Das ist tatsächlich etwas, was man mal als Marketing-Element aufnehmen könnte. Ist mir noch gar nicht in den Sinn gekommen, du hast natürlich recht. Mir war auch oft selbst mulmig, wenn ich nicht gut vorbereitet war auf ein Teachlet. Ich bin ja immer sozusagen der Meta-Moderator als Veranstalter, und mir war oft mulmig, dass ich dann dachte: Hm, ich hab das selbst nicht so richtig durchdrungen, was er da vorbereitet hat. Und dann ergibt sich das eigentlich immer, und das ist sicher auch ein wichtiges Element von den Teachlets und von der Teachlet-Werkstatt, dass da ein Klima des Vertrauens entstehen muss. Das heißt, dass alle, die da sitzen, auch das Gefühl haben, sie werden da nicht gegrillt, sondern es geht tatsächlich darum, dass alle etwas lernen. Wenn dann alle auch den Mut haben, Kritik zu äußern und Vorschläge zu machen, und wenn dieses Klima erst mal vorhanden ist, dann kann man tatsächlich auch ein schwaches Teachlet gut tragen in so einer Veranstaltung. Das ist, glaube ich, ein gutes Element.

Julian Fietkau:
Und wenn du jetzt den Teachlet-Moderatoren der Zukunft (wir wollen ja hoffen, dass das Konzept noch lange erhalten bleibt und lange eingesetzt wird) irgendwelche Hinweise geben müsstest, was würdest du den Leuten mit auf den Weg geben? Was steht noch nirgends auf Papier, was wurde noch nie aufgeschrieben, aber ist irgendwie wichtig und muss bedacht werden?

Axel Schmolitzky:
Es darf auf gar keinen Fall unterschätzt werden. Das ist, glaube ich, etwas, was leicht passieren kann, wenn man glaubt, dass man eine Choreographie hat und damit eine sehr gute Vorbereitung. Es ist trotzdem sehr, sehr anspruchsvoll, ein Teachlet zu moderieren. Ich glaube, es ist für viele Studierende schon eine Herausforderung, einen Vortrag zu halten, das heißt sich vor einer Gruppe von Kommilitonen hinzustellen, Folien zu zeigen, einen eigenen Gedanken zu zeigen, den auch flüssig und überzeugend rüberzubringen. Das gelingt schon vielen nicht so gut, entweder weil sie nicht gut vorbereitet sind oder weil sie vielleicht nicht das Talent haben, sich zu präsentieren. Das muss schon auch ein bisschen Extrovertiertheit dabei sein, damit das gut funktioniert. All das ist noch viel mehr gefordert bei einem Teachlet, weil es da eben nicht nur darum geht, den Gedanken den man sich gut vorbereitet hat, entsprechend gut vorzutragen (was man auch durch Auswendiglernen schaffen kann), sondern man muss viel, viel mehr reagieren können auf das, was da von den Teilnehmern kommt. Man muss sogar die Vorschläge ganz schnell bewerten und einordnen können. Das heißt, diesen Entwurfsraum, der sich da aufspannt durch das Teachlet, den sollte man als Moderator sehr, sehr gut beherrschen. Das ist etwas, das viele gerne unterschätzen. Das ist mein Eindruck, dass das von den Leuten, bei denen die Teachlets nicht so gut laufen, gerne unterschätzt wird. Dass sie einfach, ich würde mal vermuten, eher zu bequem waren, da wirklich mal gründlich diesen Entwurfsraum zu erkunden, indem sie einfach selber verschiedene Alternativen mal programmieren, vielleicht auch mit Leuten schon mal vorher diskutieren was es da für Möglichkeiten gibt, um einfach nicht zu sehr überrascht zu sein von den Entwurfsvorschlägen, die dann da kommen. Die Vorbereitung ist schon sehr, sehr wichtig. Ich denke, sehr viel wichtiger als bei einem normalen Vortrag.

Andererseits sollte man sich auch nicht zu sehr abschrecken lassen, also man sollte nicht zu sehr eingeschüchtert sein. Es sollte für einen Moderator auch immer, zumindest in der Teachlet-Werkstatt, klar sein, dass niemandem der Kopf abgerissen wird. Ich weiß von mir, vielleicht dass noch als Hinweis *(lacht)*, dass ich als Veranstalter sehr viel gereizter bin als Teilnehmer (ich hab ja eine gewisse Einflussnahme, weil ich als Meta-Moderator da bin, aber auch als Teilnehmer). Ich kann ein Teachlet sehr wohlwollend begleiten, ich kann aber auch sehr biestig werden. Nicht, dass ich das jetzt androhen will. Ich habe es einfach am eigenen Leib erfahren, dass ich mit der Vorbereitung oft sehr, sehr unzufrieden war. Dass ich gemerkt habe, dass die Hinweise, die ich gegeben habe, nicht aufgenommen wurden, dass einfach nicht genug Arbeit hineingeflossen ist. Dann tue ich mich sehr viel schwerer als entspannter Teilnehmer da zu sitzen und das wohlwollend zu machen, als wenn ich das Gefühl habe: Mensch, das ist zwar nicht besonders doll was die gemacht haben, aber die haben sich alle Mühe gegeben und wirklich viel Arbeit

hineingesteckt. Dann kann ich das sehr viel besser mittragen. Ich glaube, ich komme eher damit klar wenn jemand Schwierigkeiten inhaltlicher Art hat, als wenn jemand einfach zu wenig Zeit oder Leidenschaft investiert. Da komme ich nicht gut mit klar.

Julian Fietkau:
Das merken Teilnehmer ja vielleicht sowieso selbst, auch wenn sie an der Vorbereitung nicht beteiligt sind. Dann möchte man ja unterstellen, dass man merkt, wenn jemand vorne steht und wenig oder keine Zeit in die Vorbereitung investiert hat. Dass das dann von den Teilnehmern bei einem Teachlet noch stärker reflektiert wird als bei einem Vortrag.

Axel Schmolitzky:
Ich denke, das ist richtig, was du sagst. Das kommt sicherlich hinzu. Die Gruppe, wenn sie gut funktioniert, ist relativ schnell sehr empfänglich dafür, ob da jetzt jemand vorne steht, der sich gut vorbereitet hat, oder nicht. Das wird sehr schnell herausgefunden, das glaube ich auch. Klar, wenn da die Zeit von allen in Anspruch genommen wird mit etwas, was nicht gut vorbereitet ist, dann kann es auch das Feedback entsprechend geben.

Julian Fietkau:
Gibt es sonst noch irgendetwas was du loswerden möchtest zum Thema?

Axel Schmolitzky:
Was ich loswerden möchte... Ich möchte zum einen mal loswerden: Ich finde es cool, dass du diese Bachelorarbeit machst, weil es einfach spannend ist, in dieser reflektierten Form die Teachlets zu untersuchen, ein bisschen die Grenzen auszuloten.

Ich glaube, dass es ein bisschen schade ist, dass die Teachlets noch so wenig bekannt geworden sind, dass sie so wenig nach außen getragen werden. Das liegt sicherlich auch an unserer eigenen Öffentlichkeitsarbeit. Christian Späh, der auch die letzten Jahre immer gern bei der Teachlet-Werkstatt aktiv dabei war und mir da sehr viel geholfen und mich sehr stark unterstützt hat, dass wir zusammen überlegt haben: Wir müssen mal eine Webseite bauen, dass diese Webseite auch nie so richtig aus dem Alpha-Stadium herausgekommen ist.[9] Dadurch ist natürlich bisher wenig nach außen gedrungen. Ich denke, man könnte mit einer besseren Öffentlichkeitsarbeit das ganze Konzept ein bisschen besser positionieren, das wäre sicherlich möglich.

Aber es ist auch nach wie vor sehr schwierig, Teachlets Menschen nahezubringen, die nicht selbst mal an einer Teachlet-Werkstatt teilgenommen haben. Wenn man dann die paar PowerPoint-Dateien, die Ausgangssystem-Zip und Zielsystem-Zip, wenn man diese Dateien zusammenpackt und sagt: Hier ist ein Paket für ein Teachlet, und das gibt man jemandem in die Hand, der kann das schlecht einschätzen, wie gut oder wie schlecht das ist. Ein Teachlet, und das ist, glaube ich, die große Schwierigkeit von Teachlets, zeigt sich nur in seiner Durchführung. Ansonsten sind Teachlets sehr schwer zu greifen, sehr unscheinbar. Man kriegt sie nicht so richtig zu fassen. Erst in der Durchführung werden sie stark, eine Durchführung dauert aber 90 Minuten und die können die meisten nicht investieren. Es wäre also zum Beispiel mal interessant, einen Werbefilm über Teachlets zu

[9] `http://www.teachlets.org/`

drehen, könnte ich mir vorstellen. Oder, was den Teachlets sicherlich auch gut täte, wenn wir die überwiegend deutschen Teachlets auch für internationale Interessen übersetzen würden. Wir haben, glaube ich, nur ein, zwei Teachlets, die auch in Englisch zu Verfügung stehen. Alle anderen sind natürlich, weil wir ja ein deutsches Seminar durchführen, in deutsch gehalten.

Dann gibt es noch eine Randbemerkung auch zur Öffentlichkeit von Teachlets: Es gibt einen geschützten Namen *Teachlet* in den USA, deswegen bin ich auch nicht sehr offensiv gewesen bisher, diesen Teachlet-Begriff mit der Belegung weiter bekannt zu machen. Ich war da sehr vorsichtig. Dieser andere Teachlet-Begriff ist tatsächlich das genaue Gegenteil, da geht es um Lehreinheiten im Internet, also *distance teaching*, *distance education*. Und das passt gar nicht mit Face-to-Face-Interaktion in einem Klassenraum zusammen. Das hat sich auch schon mal negativ auf die Bewertung eines Papers von mir ausgewirkt. Aber das ist eine andere Geschichte.

Julian Fietkau:
Okay, das war's?

Axel Schmolitzky:
Ja, wenn du nicht noch Fragen hast? Ich hab sonst jetzt nichts, was mir noch auf der Seele brennt.

Julian Fietkau:
Ich glaube, wir hatten auch alle Themen, die mir wichtig waren. Von daher: Danke schön!

Axel Schmolitzky:
Gerne.

Interview mit Carola Lilienthal

Dieses Interview wurde am 13. September 2010 in Raum D-204 am Informatikum der Universität Hamburg durchgeführt. Es begann um 9 Uhr und dauerte ca. 20 Minuten.

Julian Fietkau:
Noch mal offiziell hallo und danke, dass du dir ein bisschen Zeit für mich genommen hast. Dann komme ich gleich mal zur ersten Frage, die auch als Einleitung hoffentlich ganz gut geeignet ist, und zwar: Welche Erfahrungen mit Teachlets hast du denn bisher gesammelt, ganz allgemein?

Carola Lilienthal:
Ich habe zuerst von Axel davon gehört, der das ja in seinem Seminar entwickelt hat. Da war ich glaube ich hier auch noch an der Uni, ich war ja bis 2008 hier als Assistentin. Und dann... Das ist gerade schwierig, wieder nachzuvollziehen, wo ich das das erste mal gemacht habe. Wir haben verschiedenste Lehrfomen aus den Firmen in die Uni und hin und her transportiert und ich hab immer wieder die Erfahrung gemacht: Wenn ich den Leuten etwas nur theoretisch erkläre, dann komme ich damit nicht weit. Wir haben zum Teil auch Kurse gemacht, in denen wir Leuten Entwurfsmuster beigebracht haben, und da hat sich das quasi aufgedrängt, dass man das auch mal ausprobiert. Ich hab das dann erst mal Kollegen von mir ausprobieren lassen, die solche Schulungen gemacht haben. Dann hab ich irgendwann an der Fachhochschule hier[1] am Berliner Tor für die Fünftsemester im Bachelor eine Vorlesung zu Architektur gehalten, und dann hab ich es auch selbst mal ausprobiert.

Ich hab das bestimmt auch anders gemacht als Axel das tut. Ich hab das zum Teil gar nicht mit Software gemacht, sondern auch mit: Wir zeichnen hier zusammen ein Diagramm oder sowas. Also dass man die Leute dazu kriegt, mitzumachen, denn ich hasse nichts mehr als wenn sie mir einschlafen bei was auch immer ich da vorne tue. *(lacht)* Insofern sind sie irgendwann ganz natürlich aufgetaucht, als Axel das gemacht hat. Ja, gute Ideen werden einfach verwendet.

Dann habe ich das eben an der Fachhochschule viel gemacht, da hatte ich aber auch, das war so meine Erfahrung im letzten Semester, so zwanzig, dreißig, vierzig Leute in einer Vorlesung. Jetzt hatte ich eigentlich vor – ich habe ja STE[2] gemacht hier an der Uni letztes Semester, den einen Teil von SE2[3] – da wollte ich auch gerne ein Teachlet machen, und da habe ich gemerkt: Mit 150 Leuten oder zumindest in einem so großen Hörsaal, auch wenn das dann weniger werden mit der Zeit, geht das eigentlich nicht. Man sitzt zu weit voneinander weg, man hört sich nicht – ich höre nicht, was die sagen, und die hören sich untereinander schon mal gar nicht – der Ort ist gar nicht geeignet dafür. Da habe ich es dann in dem Zusammenhang mal gelassen, in der Vorlesung hatte es keinen Sinn. Aber das bedeutet nicht, dass ich das nicht wieder mache. Nur eben mit so vielen Leuten... oder eben in einem Kontext, in dem man sich nicht hört, ist es einfach

[1] Hochschule für Angewandte Wissenschaften Hamburg.
[2] Softwaretechnik und Softwareergonomie, ein Teil des Moduls Softwareentwicklung 2.
[3] Softwareentwicklung 2, eine Veranstaltung im zweiten Semester der Informatik-Studiengänge der Universität Hamburg.

Unsinn. Da kann man das nicht machen.

Und ich habe es meinen Studenten an der Fachhochschule zum Teil als Aufgabe gegeben, dass sie in einer Übung – die hatten neben der Vorlesung auch eine Übung, die war nicht so ausschweifend wie das hier so ist, aber vielleicht ist das im fünften Semester auch schon wieder etwas anderes – da habe ich ihnen fertige Teachlets gegeben, die sie sich selbst aneignen und dann vorführen sollten.

Julian Fietkau:
Und wo kamen die her, die Teachlets?

Carola Lilienthal:
Die kamen hier aus Axels Seminar. Die waren also schon fertig. Er hat ja eine Webseite oder einen CommSy[4]-Raum dafür und da durfte ich mir die dann runter holen. Hab mir auch das Einverständnis der Autoren jeweils besorgt und dann haben die Studenten das ausprobiert. Das war auch spannend, mit so einer fertigen Aufgabe etwas zu machen. Das ist ja noch mal etwas anderes, als wenn du sie dir selbst ausdenkst. Wobei, ich hab mir die auch nie selbst ausgedacht, ich hab auch das genommen, was da war.

Julian Fietkau:
Und das hat also gut funktioniert?

Carola Lilienthal:
Ja, das hat gut funktioniert. Was da eher schwierig war, war, dass man für so ein Teachlet eigentlich 90 Minuten braucht, und wir hatten natürlich nicht für jede Gruppe 90 Minuten. Sie mussten das dann irgendwie kürzen, das fanden sie dann doof, und naja... *(lacht)*

Julian Fietkau:
Du kennst ja auch noch andere Lehrformen wie klassische Vorlesungen oder Übungsgruppen. Wenn du die mal vergleichen würdest mit Teachlets, wo siehst du da so die Unterschiede, Vor- und Nachteile?

Carola Lilienthal:
Gegenüber einer Vorlesung hat es auf jeden Fall Vorteile, weil ich nicht nur da vorne stehe und Wissen weitergebe und die Leute nur zuhören müssen und ich sie alleine durch meine Performance da vorne irgendwie wach halten muss, sondern dass die Leute mitmachen. Sie müssen sich melden, sie müssen sich beteiligen – das ist ein total gutes Mittel, um die Leute dazu zu bringen, dabei zu sein und auch in dem Moment sich mit dem Stoff zu beschäftigen, und nicht mir zuzuhören, mich zu verstehen, das aufzuschreiben oder nachzulesen um dann den Erkenntnisschritt zu haben. Das wäre ja viel zu spät, dann können sie mich gar nicht mehr fragen. So kommen sie da viel dichter heran.

Weil du jetzt auch nach Übungen fragtest: Die Übungen, die wir hier machen im Labor, sind natürlich noch mal effektiver, der Einzelne macht dann selbst, alleine. In dieser Teachlet-Geschichte gilt natürlich auch: Die Studenten, die sich beteiligen, sind in

[4]CommSy: an der Universität Hamburg entwickelte Software für virtuelle Projektplanung, Kommunikation und e-Learning.

der Regel immer die selben, du erreichst nicht alle. Es gibt immer noch welche, die sitzen dabei und gucken in die Gegend. Wenn die selbst vor dem Rechner sitzen würde, wäre das auch noch etwas anderes. Also es ist schon irgendwo dazwischen, aber ich glaube das Beste, was man hinbekommen kann, wenn man größere Mengen hat, die da in der Vorlesung sitzen. Wenn die alle im Labor sitzen würde, dann würde man ja auch nie mehr als Übungsaufgaben hinbekommen, eine Vorlesung kann man da gar nicht halten.

Julian Fietkau:
Jetzt hast du ja schon einige Vorteile genannt, gibt es noch weitere, die dir einfallen? Wo siehst du Kontexte, in denen sich Teachlets besonders gut einsetzen lassen, was muss dafür gegeben sein?

Carola Lilienthal:
Also einmal hab ich ja eben schon über das organisatorische Setting gesprochen, zu groß darf die Gruppe nicht sein, du musst dich hören können, dich verständigen können ohne Mikrofon, sonst wird das einfach zu zäh. Für Programmieraufgaben eignet sich das gut, also für irgendwelche programmiertechnischen Lösungen.

Ich hab dann mit den Leuten auch mal etwas anderes gemacht, ich hab gesagt: So, wir gucken uns jetzt diese Software an, als Vorbereitung auf das nächste Teachlet, und machen da ein Klassendiagramm, ein Objektdiagramm, Sequenzdiagramm... dass man sozusagen sieht: Was tut die Software? Dann haben Verfechter der Teachlets hinterher zu mir gesagt: Naja, ein echtes Teachlet war das natürlich nicht, denn das Ergebnis war ja vorbestimmt. Es gab ja gar keine Varianz, es gab nur eine im Weg, also wie man da hin kommt. Ich bin mir auch nicht so sicher, ob die Lösung beim Teachlet wirklich so offen ist, weil man natürlich als Moderator das Ganze in die Richtung lenkt, wo man hin will. Man will ja nicht, dass wenn man das *State*-Pattern den Leuten beibringt, man ein *Observer*-Pattern dabei rausbekommt. Das würde man auch verhindern. Also dafür hat sich das auch gut geeignet, das war für mich sozusagen ein Vor-Teachlet vor dem eigentlichen Teachlet. Wir üben UML-Diagramme an diesem Beispiel, was wir nachher noch weiter verarbeiten. Dafür war das natürlich ganz gut, weil die Leute dann ein Bild hatten. Ich konnte nächstes Mal sagen: Habt ihr denn noch das Sequenzdiagramm, wie war denn das, wo sind wir denn da und wo machen wir jetzt weiter? Dafür hat sich das gut geeignet.

Ich mache ja auch so Architekturanalyse, dass ich mir eine Software angucke und wie sie gebaut ist im Inneren, und das kann man auch ein bisschen damit machen. Wenn man ein Problemfall finden will und mit den Leuten eine Lösung dafür erarbeiten will, dann kann man das auch gut machen, dass man vorne sitzt und sagt: So, jetzt sagt mir mal, was ich machen soll. Ein bisschen was in der Richtung habe ich auch schon mal probiert. Da ist es natürlich nicht so, dass man sagen kann: Wir haben jetzt genau dieses Problem. Also so, wie man das bei einem Teachlet machen würde: Hier ist die Problembeschreibung, jetzt müsst ihr das mal umbauen, damit das besser wird. Sondern es geht da mehr darum, die Struktur zu erkennen und sich da durch zu finden.

Aber ich bin da vielleicht ein bisschen pragmatisch. Für mich ist das eine Technik, um Leute zu aktivieren. Was ich da nun mache oder ob ich mich an genaue Vorschriften oder

Formen halte... Da denke ich dann, muss ja nicht sein. Insofern würde jemand, der das sehr formal sieht, vielleicht sagen: Das war doch gar kein Teachlet, was du da gemacht hast. Aber für mich fällt das immer wieder darunter. Wenn ich die Leute aktiviere, dass die mir sagen, was ich tun soll, und ich moderiere das so, dass wir zu irgendeinem Ziel kommen, dann finde ich das schon sehr Teachlet-artig. So ordne ich das dann ein. *(lacht)*

Julian Fietkau:
Wozu du ja auch schon ein bisschen was gesagt hattest: Wo siehst du noch Grenzen des Konzepts? Wir hatten schon ein paar mal über die Gruppengröße gesprochen, gibt es noch andere Kontexte wo du sagen würdest: Da lassen sich Teachlets auf keinen Fall oder nur ganz schlecht einsetzen?

Carola Lilienthal:
Also Lautstärke und sowas... aber ansonsten kann ich mir fast nichts vorstellen. Wenn ich irgendwelche mathematischen Dinge den Leuten verklickern will oder irgendwelche Algorithmen, dann wird es auch so sein, dass ich versuchen kann, sie zu aktivieren, wenn ich das schlau mache. Aber das ist eben ein unheimlicher Aufwand. Das merkt man ja auch: Ein wirklich gutes Teachlet ist schwierig zu machen. Mit jeder Art von Wissen, was ich vermitteln will, wird es wieder so sein: Ich muss mich echt anstrengen um das so aufzubauen, dass die Leute eine Idee bekommen, was wir da wollen, und dann auch den Weg finden. Ich könnte mir auch ein chemisches Experiment vorstellen, wo man das macht. *(lacht)* Das kann man, glaube ich, bei vielen Sachen machen, nicht nur in der Informatik. Ich kann mir eigentlich gar keine anderen Probleme vorstellen. Na klar, wenn man etwas viel zu schwieriges macht für Leute, die noch nie vor einem Rechner gesessen haben, sowas geht natürlich nicht. Es muss immer angemessen sein. Aber das wäre ja auch eine Karikatur dessen, was wir hier wollen. Auf die Idee würde ich gar nicht kommen. Insofern: Wenn die Leute das prinzipiell können könnten, dann geht das.

Wo man es wahrscheinlich nicht machen kann, ist, wenn man einen Artikel vorstellt, also was jemand geschrieben hat, wenn man das wiedergeben will. Sonst müsste man die Leute dazu bekommen, dass sie quasi mit einem den Artikel neu schreiben. Aber vielleicht geht das auch? *(lacht)* Müssen wir Axel mal fragen, ob er das nicht schon mal ausprobiert hat. Er hat ja noch dieses Klassiker-Seminar, ob man da ein Teachlet draus machen könnte. Vieles ist möglich, also insofern: Nö, eigentlich hab ich da keine Bedenken. Man kann es bei allem probieren.

Julian Fietkau:
Hast du noch für Teachlet-Moderatoren, die das noch vor sich haben, oder Leute, die das jetzt machen, irgendwelche Tipps? Irgendwas wo du sagen wirdest, das ist vielleicht noch nirgends aufgeschrieben, ist aber total wichtig beim Moderieren von Teachlets?

Carola Lilienthal:
Also ich weiß erst mal nicht, was überall irgendwo aufgeschrieben ist. *(lacht)* Was ich glaube, was unheimlich wichtig ist, ist, dass man sich vorher noch mal klar macht: Was wollte ich hier eigentlich mit den Leuten machen? In einem Semester hatte ich das Teachlet gemacht und im Semester danach wollte ich es wieder machen, und ich

habe gemerkt: Okay, ich muss die Aufgabe selbst noch mal durchmachen, sonst geht das schief, sonst weiß ich auch gar nicht, was ich hier wollte. Man muss als Moderator sehr, sehr drin sein in dem Ding, was man macht. Sonst wirkt man auch unsicher vorne. Man muss wissen, was das für eine Aufgabe ist, aus was für Teilen sie besteht und so. Das ist natürlich eine Grundsätzlichkeit, die du immer hast, wenn du etwas vorträgst, aber gerade die Denkweise: Ach ja, den Sourcecode und das alles kenne ich ja, da wird es dann gefährlich, da muss man aufpassen.

Und ansonsten muss man gewillt sein, sich auch zurückzunehmen und die Leute kommen zu lassen, das ist oft auch nicht so einfach. Mir fällt es häufig schwer: Das ist doch offensichtlich, das müsst ihr doch sehen, in welche Richtung man gehen muss... Das ist noch wichtig, dass man sich wirklich eine Moderationsrolle anzieht. Ich muss mal überlegen, was ich sonst immer so gemacht hab... Bei einem Teachlet, das ich hatte, wurde das Lösungsmuster schon vorher vorgestellt und dann wurde das gebaut. Ich finde, darüber kann man sich auch streiten, ob man das so macht. Manchmal ist es vielleicht gar nicht schlecht, das erst mal gar nicht zu zeigen und zu sagen: Okay, wie lösen wir das jetzt? Das ist natürlich ein längerer Prozess, weil die Leute dann anfangen zu diskutieren, wie man das gut machen kann vom Design her. Aber dann lernen die Leute noch mehr, weil sie dann selbst irgendwann diese Idee haben, was eine gute Lösung wäre.

Julian Fietkau:
Ich habe schon beides erlebt. Ich war in dem Seminar letztes Jahr dabei und habe dort den Eindruck gewonnen, dass es auf das Muster ankommt. Es gibt Muster, bei denen das mit Bachelor-Viertsemestern klappt, die das dann hinbekommen das Muster sozusagen selbst zu entwickeln. Aber es gibt, glaube ich, auch welche, die dafür zu komplex sind.

Carola Lilienthal:
Ja, das kann ich mir auch vorstellen. Gerade solche wie Beobachter mit Vererbung und allem drum und dran, das ist wahrscheinlich zu schwierig. Da muss mal auch immer gucken, welche Seite sich eignet. Eine gute Vorbereitung ist da auch alles. Wenn man das gut macht, dann wird es schön. *(lacht)*

Was ich manchmal gemacht habe, ist, den Leuten vorher einen Artikel zu lesen zu geben, sozusagen als Vorbereitung auf das Teachlet. Das fand ich auch manchmal noch ganz gut. Die Leute, die sich engagieren wollten, die hatten dann vorher etwas. Das waren zwei oder drei Seiten, also gar nichts langes. Es gibt solche Kolumnen in manchen Zeitschriften, wo dann, keine Ahnung, der Fowler[5] zwei Seiten über Schnittstellen schreibt, also wie man gute Schnittstellen baut. Wenn du dann das *Facade*-Pattern verwenden oder einführen willst, dann eignet sich das ganz gut. Dann haben sie schon ein bisschen in der Richtung nachgedacht. Sowas habe ich manchmal zur Vorbereitung gemacht und fand das auch ganz spannend zu sehen, wie sie damit darauf reagiert haben. Das weiß ich nicht, ob das schon irgendwo aufgeschrieben ist. Das könnte man vielleicht noch mal versuchen, auch in anderen Kontexten.

[5] Martin Fowler, Informatiker mit Spezialisierung auf Softwarearchitektur und objektorientierte Analyse und Design, Autor. http://martinfowler.com/

Julian Fietkau:
Dann bin ich mit meinen Fragen auch schon wieder am Ende. Hast du sonst noch irgendetwas, was du zum Thema sagen möchtest, was ich noch aufnehmen und eventuell verwerten kann?

Carola Lilienthal:
Das kann man sicherlich nicht verwerten, aber es würde mich interessieren: Es gibt ja bestimmt viel theoretisches Wissen oder Wissenschaft über interaktive Lehrformen, wie sich das mit anderen verhält und was es sonst noch so gibt. Darauf habe ich gar keinen Blick. Wir machen hier unsere Laborarbeit ja wenigstens etwas interaktiver, in Vorlesungen versuchen wir, ein bisschen was mit Teachlets zu machen. Aber ich vermute, dass die Pädagogik auch noch ihre Ideen hat. Das fände ich ganz spannend zu wissen, was es sonst noch so alles gibt. Das ist jetzt kein Hinweis in Richtung Teachlets für dich, aber das könnte ja vielleicht in deiner Arbeit stehen. *(lacht)*

Julian Fietkau:
Zumindest ansatzweise, ja. Es wird mit überlegt.

Carola Lilienthal:
Das fände ich spannend. Dann könnte man das besser einordnen. Wenn man Teachlets so gewohnt ist und das Konzept schon so kennt wie ich, dann denkt man: Ist ja ganz normal, das machen die Pädagogen bestimmt auch. Ich kann mir gar nicht vorstellen, dass die sowas nicht machen. Ich denke dann an meinen Sohn in der Schule, die Lehrer versuchen natürlich schon immer, irgendwas zu machen, damit die Kinder mitmachen. Die wissen genau, dass es nichts nützt, denen da vorne etwas vorzuerzählen. So ist das natürlich eigentlich in einer Vorlesung auch besser. Theoretisch, denke ich, müssten die noch mehr interaktive Lehrformen haben, auf die man verweisen kann. Das würde mich noch interessieren. Ansonsten... bin ich ein Fan. *(lacht)* Ich hoffe, ich habe dir ein paar spannende Antworten gegeben oder irgendwas gesagt, was wichtig war.

Julian Fietkau:
Auf jeden Fall. Dann danke sehr und wir gucken mal, was daraus wird.

Carola Lilienthal:
Gern geschehen.

Interview mit Christian Späh

Dieses Interview wurde am 13. September 2010 in Raum D-214 am Informatikum der Universität Hamburg durchgeführt. Es begann um 11 Uhr und dauerte ca. 30 Minuten.

Julian Fietkau:
Dann noch mal hallo Christian, vielleicht magst du dich bitte ein mal kurz vorstellen?

Christian Späh:
Ich bin Christian Späh, wissenschaftlicher Mitarbeiter am Arbeitsbereich Softwaretechnik, und habe zusammen mit Axel Schmolitzky jetzt schon mehrmals die Teachlet-Werkstatt durchgeführt und auch ein mal komplett alleine. Und jetzt sitze ich hier und beantworte deine Fragen. *(lacht)*

Julian Fietkau:
Gut, dann kommen wir doch gleich mal zur ersten Frage, und zwar: Ganz allgemein, welche Erfahrungen mit Teachlets hast du denn bisher so gesammelt?

Christian Späh:
Oh, die ist aber sehr allgemein. *(lacht)* Als erstes: überaus positive! Ich hab sie ja selbst als Student auch besucht und habe bei keiner anderen Veranstaltung damals so viel über Softwaretechnik verstanden wie in dieser Teachlet-Werkstatt. Ich habe damals zum ersten mal richtig verstanden, wie man Referenzen einsetzt – damals waren SE1[1] und SE2[2] noch nicht ganz so strukturiert wie jetzt, vielleicht lag es auch an mir, dass ich es nicht verstanden habe während der Vorlesung.

Ich fand es immer wieder interessant auch die anderen Teachlets zu sehen und mich mit den Themen auseinanderzusetzen. Es fällt mir selbst immer sehr schwer, anhand von Büchern über Entwurfsmuster zu reden, aber anhand von Softwarebeispielen finde ich es dann doch sehr leicht. Auch sehr aufwändig, das habe ich dann festgestellt bei der Betreuung und der Vorbereitung. Es ist nicht so wie in einem Seminar, wo man als Dozent fast immer erst mal Themen vorgeben kann und dann lässt man die Studierenden einfach laufen, dann wird das schon irgendwie gut gehen. Wenn man nicht so viel Zeit investieren will, dann halten sie einfach ihren Vortrag und gut ist. So war es hauptsächlich während meines eigenen Studiums mit den Seminaren, die ich hier besucht habe. Wenn die Teachlet-Werkstatt gut werden soll, muss man sich als Veranstalter ziemlich viel Zeit nehmen. Ich hatte mir damals immer zwei Termine vorher ausgesucht, bei denen ich mit den Studierenden die Teachlets bespreche, die sie vorbereiten, und hab dann nach den ersten beiden Teachlets gemerkt: Das ist viel zu wenig, auch mit zwei Wochen vorher. Dann habe ich das weiter nach vorne gezogen und mich drei Wochen vorher und auch drei mal mit den Leuten getroffen. Das dauert dann auch immer mindestens eine Stunde oder länger. Dann hat mal quasi innerhalb von drei Wochen immer drei Termine, die jeweils eine Stunde dauern, plus die Teachlet-Werkstatt, die auch noch mal drei Stunden

[1] Softwareentwicklung 1, eine Veranstaltung im ersten Semester der Informatik-Studiengänge der Universität Hamburg.
[2] Softwareentwicklung 2, eine Veranstaltung im zweiten Semester der Informatik-Studiengänge der Universität Hamburg.

dauert, das ist schon eine ziemlich ordentliche Zeitfressmaschine. Aber nur dann haben zumindest Axel und ich – ich weiß nicht, was Axel dir noch so berichtet – das Gefühl, dass die Teachlets so gut sind, dass man sie auch präsentieren kann.

Man hat es da ja zum großen Teil mit Studierenden zu tun, für die Entwurfsmuster neu sind, die sich damit noch nicht so gut auskennen, wo wir also nur vom Erfahrungsstand von SE1 und SE2 ausgehen können. Das heißt sie haben also noch keine große Software gebaut und es fällt ihnen zum Teil schwer, diese Software so zu bauen, dass sie für ein Teachlet geeignet ist.

Es zählt aber nicht nur die Erfahrung, dass man schon mal Software gebaut hat, sondern auch die Teachlet-Erfahrung ist bei den Teilnehmern ja noch überhaupt nicht vorhanden, die kommen ja selbst auch zum ersten mal damit in Kontakt. Da machen Axel und ich die Erfahrung, dass im Grunde erst drei Jahre Erfahrung mit Teachlets wirklich dazu befähigen, gute Teachlets zu entwickeln. Es gibt zwar auch Ausnahmen, so wie dich *(lacht)*, aber worauf es dann so ankommt, wie die Software aussehen soll, da gibt es viele Feinheiten, auf die man zwar immer wieder hinweisen kann. Aber oftmals verstehen sie es dann nicht oder können es nicht so umsetzen.

Ja, was sind so meine Erfahrungen mit Teachlets... Für Entwurfsmuster sind sie hervorragend geeignet. Wie ein Teachlet abläuft, entscheidet sich wirklich durch die Vortragenden live vor Ort. Das Teachlet kann in der Vorbereitung den Eindruck gemacht haben es ist miserabel, und das Teachlet läuft dann trotzdem gut ab. A, weil der- oder diejenigen dann doch das Ganze gut verstanden haben und gut präsentieren können, auch wenn sie es schlecht vorbereitet haben. B, weil die Teilnehmer sehr gut mitmachen. Entweder interessiert sie das Thema und sie sind sehr gut dabei, oder die Ausgangssoftware ist dann doch so spannend für sie, oder aber sie haben teilweise auch das Gefühl: Wir müssen dem da vorne ja mal helfen, also müssen wir uns jetzt beteiligen, und bringen dann quasi selbst das Teachlet voran, auch wenn der- oder diejenige aus meiner Sicht dann vorne eigentlich eher schon fast am Untergehen ist. Aber trotzdem kann das Teachlet dann gut sein.

Womit hatte ich gerade angefangen? Ja, und Teachlets, wo man eigentlich bei der Vorbereitung das Gefühl hatte: Das läuft sehr gut, sind dann bei der Durchführung, ich sag's mal so drastisch, miserabel. Weil dann doch gerade die letzten Änderungen, die man vorgeschlagen hat, die unheimlich wichtig sind und wo man dachte, diejenigen hätten das verstanden und würden das noch einbauen, die Leute dann so schlusig waren oder das nicht so reingearbeitet haben, dass das Teachlet dadurch einfach miserabel war. Sie haben dann bestimmte Sachen in der Choreographie nicht bedacht, denn das ist ja auch ein unheimlich wichtiger Bestandteil. Bei einigen Teachlets hängt es wirklich von der Aufgabenstellung ab, wann man etwas sagt und wie ausführlich man etwas sagt, damit die Aufgabe ganz konkret rüberkommt und auch sinnhaft ist, und nicht einfach so: Wir machen das jetzt mal, weil wir's machen wollen, sondern weil wirklich eine Anforderung da ist, die es erfordert, dass wir das so bauen. Wir hatten da echt schon alles, sowohl in die eine Richtung als auch in die andere.

Meine Erfahrung mit Teachlets... Axel und ich machen zu Beginn der Teachlet-Werkstatt die eigene Erfahrung mit unseren Demo-Teachlets, die wir halten, dass allein das Vorbereiten und das Durchführen eines Teachlets, das schon ausgearbeitet ist, auch

unheimlich anstrengend für uns ist. Wenn man ein Dreivierteljahr mit den Teachlets nichts mehr zu tun gehabt hat – denn das findet ja nur ein mal im Jahr statt – da erst mal wieder reinkommen, sich wieder alles in den Kopf zurückholen worauf man achten möchte, und sich vorne hinsetzen und dann wirklich das Teachlet vorführen, und was ja da immer so anstrengend ist, die Diskussionen leiten, zu einem Ziel führen und trotzdem immer noch den Überblick behalten, dass auch alles, was von den Teilnehmern kommt, auch bedacht oder zumindest mal angesprochen wird, das ist unheimlich anstrengend. So dass wir am Anfang die ersten zwei Termine eigentlich immer denken: Oh Gott, was machen wir da wieder, das ist ja so zeitaufwändig, man kann sich da gar nicht überall so reinarbeiten. Aber spätestens nach dem dritten Termin, wo dann die selbstgemachten Teachlets der Studierenden kommen, macht das dann wieder so richtig Spaß, weil man dann merkt wie gut die das vorne präsentieren und wie gut das bei den Teilnehmern ankommt.

Julian Fietkau:
Wie viele Teachlets hast du denn schon überlebt, so über den Daumen gepeilt?

Christian Späh:
Wir haben in einem Seminar an neun Terminen maximal Teachlets. Ich habe das das eine Jahr komplett selbst gemacht, da sind es schon mal neun. Ich war ein mal als Student selbst dabei, dann sind es 18. Dazwischen waren ungefähr vier Jahre, wo ich mal mehr, mal weniger daran beteiligt war. Ich hab auf keinen Fall immer alle neun gesehen. Aber sagen wir, da waren es auch noch mal 18. Also 36, würde ich sagen.

Julian Fietkau:
Das hilft sicher dabei, deine Erfahrung einzuordnen, da du ja – abgesehen von Axel – die Person bist, die da bisher am meisten miterlebt hast.

Christian Späh:
Ja, das erscheint mir auch so.

Julian Fietkau:
Und wenn du jetzt Teachlets vergleichst mit anderen Lehrformen? Du hast ja als ehemaliger Student auch Erfahrung mit Vorlesungen, du leitest Übungsgruppen... Wo siehst du da so die Unterschiede? Wofür sind Teachlets gut geeignet?

Christian Späh:
Du weißt, da stehe ich immer in Diskussion mit Axel, also ich werde da sicherlich nicht viel Anderes erzählen. *(lacht)* Teachlets sind einfach da sehr gut geeignet, wo es darauf ankommt, Sachverhalte praktisch zu erklären. Softwareentwicklung ist so komplex, schon wenn man nur zwei oder drei Klassen hat, kann man aus meiner Sicht darüber nicht mehr anhand von Folien reden, zumindest wenn es um neue Inhalte innerhalb der Lehre geht. Wenn man eine Architektur präsentieren will und man hat da erfahrene Softwareentwickler, die das Projekt auch kennen, dann kann man das alles ohne Teachlet machen. Dann zeigt man einfach Quelltextbeispiele und eine Architektur auf einem bestimmten Niveau, bestimmte Details. Dann haben die alle genug Erfahrung und können sich dar-

über unterhalten. Aber wenn die Studierenden gerade erst herangeführt werden und die ersten Klassenstrukturen mal selber bauen, die eine bestimmte, sinnhafte Aufteilung von Funktionalität haben sollen, dann kann man denen das nicht einfach so präsentieren.

Das fängt schon bei dem Beobachtermuster an. Wenn Guido[3] das nur in seinen Folien in seiner Vorlesung präsentiert, dann können das im Grunde nur diejenigen verstehen, die schon selbst in verschiedenen Programmiersprachen Programme geschrieben haben und das einfach schon drauf haben und darunter etwas verstehen können. Aus meiner Sicht ist es dann aber so, dass die anderen noch verstehen: Okay, Beobachter, es gibt irgendjemanden, der sagt etwas, und irgendjemanden, der hört zu. Aber die können das nicht mehr in Java-Klassen überführen. Dazu fehlt ihnen einfach die Kenntnis, der Erfahrungspool, dass sie sagen können: Ich brauche jetzt eine Klasse mit der und der Schnittstelle, und ein Interface so und so, und das kann sich dann so darüber anmelden und wird dann informiert. Also das Abstrakte ist eventuell sogar noch da. Gefühlt bei einem Drittel ist selbst das nicht da, weil sie es auf diesem abstrakten Niveau einfach nicht verstehen können, und das eine Drittel von Leuten, die das noch abstrakt verstehen, kann es nicht mehr überführen. Meine pauschale Aussage ist: Zwei Dritteln würde die Vorlesung nichts bringen, wenn man das nicht noch irgendwie eintrainiert.

In SE1 und SE2 machen wir das ja mit der Übung und greifen das da noch mal auf. Das ist die eine Variante, die bei so einem Massenbetrieb wohl auch nur möglich ist. Die andere Idee ist, dass man das schon während man sagt, um was es hier geht, eine Software vorbereitet hat und die dann zusammen mit den Studierenden weiterentwickelt.

Deine konkrete Frage war, wo man das gut einsetzen kann... Also bei der Softwareentwicklung wenn es um Entwurfsmuster geht. Über alles Andere kann man nur spekulieren, und das haben wir ja auch. Ich wiederhole mich oder uns da vielleicht noch mal ein bisschen. *(JF: Das macht ja nichts.)* Mir fehlen die Erfahrungen für andere Bereiche der Softwareentwicklung, was das Hardware-mäßige angeht, was die formale Informatik angeht, wo es quasi mathematisch ist, da kann ich mir vorstellen, dass es nicht so gut funktioniert. Da ist es eher so: So oder so ist das, und jetzt wende das noch mal an.

Dann weiß ich ja, dass du auch ausprobieren wolltest und mal so den Finger vorgestreckt hast, wie man das im größeren Rahmen machen kann. Da denke ich, dass es möglich ist, das auch in einem Vorlesungsumfang zu machen. Greifen wir das Beobachtermuster wieder auf. Da denke ich, man könnte das auch so in der Art einführen, dass man sofort vor Ort die Software live weiterentwickelt. Aber ich denke, dass da die Einschränkungen sind: Es werden sich nur fünf, vielleicht maximal zehn, an der Diskussion beteiligen können, was wir ja eigentlich als essenziell bei einer Teachlet-Durchführung ansehen, dass das so aus den Teilnehmern selbst kommt. Aber ich denke, alleine das würde schon mehr bringen. Ob die anderen hundert, dreihundert Teilnehmer da nur dem Dozenten zuhören oder sich durch ihre Kommilitonen noch bereichern lassen... da, glaube ich, ist das immer noch spannender, sich das von den eigenen Kommilitonen noch etwas anzuhören, wie die sich das vorstellen, und dafür verschiedene Meinungen im Kopf mit durchzugehen. Und das fördert die Aufmerksamkeitsspanne bei den Studierenden.

[3] Dr. Guido Gryczan, Vertretungsprofessor am Arbeitsbereich Softwaretechnik und einer der Dozenten von SE2.

Wenn sie denn an Programmierung Interesse haben und dann sehen, dass live gecodet wird, dann ist das natürlich etwas sehr Interessantes.

Natürlich kann man immer sagen: Einige interessiert das überhaupt nicht, aber es ist nun mal eine Softwareenwtwicklungsveranstaltung *(lacht)* und es wird sicher auch gut ankommen. Ich war auch mal auf einer Microsoft Student Conference, da haben sich auch erfahrene Programmierer einfach mal herangesetzt und die neueste Entwicklungsumgebung für .NET, oder was das damals war, gezeigt. Die haben sich einfach hingesetzt und quasi rumgealbert. Die hatten zwar ein grobes Ziel, was sie programmieren wollten. Sie hatten sich natürlich auch die Schritte vorher ausgesucht, und was denn die Entwicklungsumgebung alles supertolles kann. Das ist aber alles wirklich erst während dieser halben, dreiviertel Stunde entstanden, was sie wirklich genau machen und mit welchen Feinheiten. Es gab natürlich schon eine gewisse Vorselektion, weil es halt so eine Microsoft Student Conference war, wer daran teilnimmt. Aber es war interessant zu sehen: Wie programmieren Andere live? Um sich da noch mal etwas abzuschauen. Ich glaube das ist ein Effekt, der dann auch in einer Vorlesung mit eine Rolle spielt, was ja anscheinend auch Axel als Erfahrung bei sich mitnimmt, wenn er in BlueJ live programmiert, um dann irgendwas zu zeigen, bei ihm in SE1 ja hauptsächlich irgendwelche Schleifenkonstrukte oder so.

Julian Fietkau:
Und wo siehst du so ungefähr die Grenzen des Konzepts? Gibt es Kontexte, wo du sagen würdest, da ergeben Teachlets überhaupt keinen Sinn?

Christian Späh:
Erst mal die allgemeine Aussage: In einem Kontext, in dem es für eine Lösung keiner Diskussion bedarf, macht es keinen Sinn. Das hatten wir ja auch bei unserem Brainstorming so herausgefunden. Ist das Ergebnis so trivial, dass man es so herunterprogrammieren kann, dann ist es sinnlos.

Das heißt, bei Softwareentwicklung muss es eine bestimmte Komplexität erreichen. Aber die erreicht man, glaube ich, schon ziemlich schnell. Alles, was mehr als „eine Schleife programmieren“ ist, kann ein Teachlet rechtfertigen. Wenn es darum geht, Methoden aufzuteilen oder so. Das kann ich mir alles in einem Teachlet vorstellen. Im Augenblick machen wir es ja so, dass es immer ein, maximal zwei größere Themen sind. Ich kann mir auch vorstellen, dass man sie kleiner gestückelt macht, quasi wirklich in einer Art von Lehrgang. Dann kommt es noch mehr auf den Vortragenden an, wie er das Ganze gestaltet.

Wir hatten ja auch Teachlets, in denen andere Sprachen vorgestellt wurden. Die Vorstellung selbst war nicht so interaktiv, da wurde erst mal Basisqualifikation vermittelt. Nachher ist das eigentliche Teachlet dann meist zu kurz gekommen, aus Axels und meiner Sicht. Das Ganze ist aber trotzdem sehr gut bei den Teilnehmern angekommen, weil sie Interesse an dieser neuen Sprache hatten. Aber da muss man dann auch gucken, ob die Teachlet-Merkmale noch eingehalten sind. Man kann es dort nicht verwenden, wo man nicht herumprobieren kann. Ich versuche, mir da irgendwas vorzustellen...

Julian Fietkau:
Du hattest vorhin etwas von Hardware gesagt. Man kann ja wahrscheinlich schlecht vorne schnell einfach mal ein bisschen herumlöten. Naja, vielleicht geht es in einem sehr begrenzten Rahmen, aber wenn du sowas wie Hardwaredesign hast?

Christian Späh:
Hardwaredesign, oder als Systemadministrator einen Fileserver aufsetzen... Im Grunde musst du dir da vorher Gedanken machen, du kannst das maximal irgendwie aufzeichnen, was greift wie worauf zu, welche Festplattencontroller brauchst du, welches RAID willst du benutzen, welche Arten von Festplatten benutzt du... Aber letztendlich musst du dir das anlesen und dann aufmalen und kannst mit den Anderen dann über das Modell reden. Aber bei der Teachlet-Werkstatt ist es ja essenziell, dass du nicht das Modell bearbeitest, sondern die lauffähige Software. Sobald du das nicht kannst, muss man noch mal diskutieren, ob das noch ein Teachlet ist. Ich weiß nicht, ob wir das beim Brainstorming so herausgearbeitet haben. Es geht nicht darum, dass man das Modell bearbeitet, sondern man braucht wirklich etwas Lauffähiges. Doch, das hatten wir, oder? Lauffähig, irgendwie sowas. Sobald man Hardware hat, man kann halt nicht einfach alles sich irgendwie bestellen und dann herumprobieren und dann die Teilnehmer in einer Art von Lehrgang einfach mal herumexperimentieren lassen: Jetzt baut mal das und baut mal das und dann testen wir mal aus, welcher RAID-Aufbau schneller ist.

Aber es ist halt auch sehr abstrakt jetzt. Man denkt sich das einfach irgendwie aus. Ob man das nicht doch irgendiwe machen kann, in irgendeiner Form... Was hätten wir denn noch? Psychologie, die Seminare die ich da hatte, da wüsste ich auch nicht, wie man Teachlets einsetzen würde. Da hatten wir auch teilweise Praxisbeispiele, Erfahrungen: Es gibt jetzt quasi diesen Versuchsaufbau und den wollten wir ein mal durchspielen. Also wie testet man jetzt eine Fahrstuhl, ob der benutzerfreundlich ist oder so. Da ist die Aufgabe klar, da ist das Szenario umrissen, du hast ein Vorgehensmodell und dann wird das durchgeführt. Also eine Übung, mit der man das einübt. Aber ich wüsste nicht, wie man da zum Beispiel jetzt ein Teachlet durchführen wollte, wo man darüber diskutiert, wie man das jetzt am besten macht, und dann führt man das durch. Man kann ja nicht alle Vorgehensmodelle wie man das testet in den Raum stellen und darüber diskutieren, welches man jetzt nimmt. Entweder du hast erfahrene Tester da und die diskutieren dann darüber, aber die kennen das dann alles schon. Aber wenn du Studierende hast, denen musst du es ja erst mal beibringen. Ich glaube, da gibt es nicht diese Form der Komplexitätssteigerung, man hat einen Basisschatz und dann kommen immer neue Sachen hinzu, größere Dimensionen in denen man denken muss um die Software zu bauen. Vielleicht ist das etwas Einmaliges in der Softwareentwicklung, dass man diese Komplexitätssteigerung wirklich immer so explizit hat und diesen fachlichen, manipulierbaren, entwickelbaren Gegenstand hat.

Auch wenn ich an die Lehrerausbildung denke, Pädagogik, oder Geschichtswissenschaften. Da geht es ja immer nur mehr um reine Wissensvermittlung im schlimmsten Fall, wo es sich jetzt hinentwickelt mit dem Bachelor, im besten Fall um philosophische Grundfragen, die diskutiert werden, wo es darauf ankommt, dass die Studierenden ihren Hirnkasten anschmeißen und dann darüber diskutieren und sich eine Meinung bilden.

Da wüsste ich auch nicht, wie man das als Teachlet machen kann. Da kannst du ja nicht auch einfach mal eine Klasse nehmen: Hier hast du zwanzig Schüler, das ist die Aufgabe, wie machst du das jetzt am besten? *(lacht)* Das kannst du da auch nicht umsetzen.

Julian Fietkau:
Wenn du jetzt den Teachlet-Moderatoren der Zukunft etwas auf den Weg geben könntest, was würdest du denen sagen? Gibt es etwas, das dir wichtig ist, dass eventuell noch nirgendwo aufgeschrieben ist?

Christian Späh:
Moderatoren im Sinne von Teachlets durchführen, nicht Teachlet-Werkstatt?

Julian Fietkau:
Genau.

Christian Späh:
Naja, ich glaube es klingt jetzt abgedroschen, ich zähle aber einfach mal auf. *(lacht)* Zeit nehmen für die Vorbereitung, es kostet leider mehr Zeit als gedacht. Axel auch und ich insbesondere hatten immer eigentlich den Gedanken, dass Teachlets sehr schnell aufgeführt werden können, wenn sie ein mal produziert wurden. Dass man sie einfach hernimmt, Axel hatte ja mal das „Teachlet in a box“ oder so, man macht die Box auf, nimmt sich das raus, guckt sich das eine halbe Stunde oder eine Stunde an und dann führt man das vor. Das funktioniert leider nicht. Die Teachlets müssen immer ein mal ausgepackt werden, ein mal durchgespielt werden, man muss sich den Quelltext ganz genau anschauen, denn es ist halt Softwareentwicklung, es kommt auf viele Feinheiten an. Man muss den Weg kennen, wie komme ich von A nach B, und man muss sich auch mit der Choreographie sehr gut auskennen, um das Teachlet sinnvoll durchzuführen. Es reicht nicht, einfach nur die Software zu nehmen.

Ja, „mit auf den Weg geben“ heißt ja auch etwas Positives benennen, also: Auch wenn man am Anfang vielleicht ein bisschen Angst davor hat, aber sich darauf einlassen sich da vorne hinzusetzen, das Ganze zu moderieren und sich auf die Diskussion einzulassen und da dann etwas rauszuziehen, das ist ein unheimlicher Erfahrungsschatz, den man da mitnehmen kann. Spannend sind auch immer die Teachlets, wo man nicht das baut, was der Moderator sich am Anfang ausgedacht hat oder was er übernommen hat. Gerade beim Befehlsmuster, was wir jedes Jahr immer wieder aufführen, haben wir noch nie dieselbe Lösung gehabt, weil es immer wieder Unterschiede gibt. Und wenn man sich auf die einlässt, dann ist es auch für die Teilnehmer wesentlich interessanter.

Mit auf den Weg geben... die ganzen technischen Details muss ich jetzt nicht aufzählen... Ganz wichtig ist, das Teachlet am Ende in einer Form zu hinterlassen, dass Andere das wieder benutzen können. Das ist unheimlich schwierig für uns zu kontrollieren, weil es aufwändig ist. Wir haben da mittlerweile ein sehr formales Muster entwickelt, das leider nicht so eingehalten wird, weil viele Studierende dafür keinen Blick haben und dann sagen: Ich muss nur die nötigsten Informationen aus dem Abstract noch ein mal reintun und das war's dann. Aber wenn es genau so wie die Vorlage ist, die wir hereingeben, dann wäre das für diejenigen, die sich das anschauen, wesentlich einfacher zu erfassen.

Also auch wirklich dieselbe Formatierung, dass man sich nicht immer umstellen muss, die Informationen suchen muss, sondern man hat ein mal das Layout, was steht wo, was ist fett, was ist nicht fett, idealerweise noch ein Bild, wie sieht die Software aus, finde ich die spannend, interessiert die mich... Das wäre schon mal ein großer Vorteil, wenn das so ins CommSy gestellt wird. Aber da haben wir ja auch jetzt gerade die andere Bachelorarbeit am Laufen, die sich da vielleicht etwas Neues ausdenkt.

Für diejenigen, die ein Teachlet neu entwickeln: Nehmt euch Zeit dafür. *(lacht)* ihr solltet mindestens drei Wochen vorher die ersten Ideen haben. Und nehmt euch das zu Herzen, was Axel und ich euch sagen, denn meistens wird es dadurch besser. *(lacht)*

Julian Fietkau:
Hast du sonst noch etwas, was du gerne hinzufügen möchtest? Etwas zum Thema Teachlets, was dir wichtig ist oder was vielleicht noch für die Arbeit von mir wichtig sein könnte, was dir noch einfällt?

Christian Späh:
Bei deiner Arbeit würde mich noch interessieren, versuchst du auch gegenüberzustellen, was Teachlet im Kleinen und Teachlet im Großen bedeuten kann oder ob das möglich ist, also von der Teilnehmeranzahl her?

Julian Fietkau:
Auch, ja. Das ist auf jeden Fall eins meiner Themen.

Christian Späh:
Und wie versuchst du jetzt, „Teachlet im Großen“ zu erfassen, wo das leider mit der Vorlesung nicht geklappt hat?

Julian Fietkau:
Es gab jetzt noch ein Teachlet beim iPhone-PowerDay, das war ein gelungenes Positivbeispiel aus meiner Sicht. Dort saßen ja auch 150 Leute im Vorlesungssaal, wo ein Teachlet gehalten wurde mit Xcode für iPhone-Entwicklung, das funktioniert hat, wo ein paar Sachen anders gelaufen sind, aber dazu hab ich schon ein bisschen was geschrieben. Aber es ging mir hauptsächlich darum, ob du noch etwas sagen möchtest. *(lacht)*

Christian Späh:
Das hat mich nur interessiert, ob ich dazu noch etwas sagen kann, aber du hast da ja noch ein ideales Beispiel gefunden, was mich sehr freut. Nein, so spontan hab ich dann nichts mehr. Ich fand das Brainstorming sehr gut. Ich glaube, da haben wir wirklich den Kern des Teachlets herausgearbeitet und auch viel, was am Rande sozusagen an Stellgrößen ist. Das hat, glaube ich, eine ganze Menge gebracht. Dazu würde ich jetzt nichts Neues sagen, was wir da nicht schon besprochen haben.

Julian Fietkau:
Okay, dann vielen Dank.

Christian Späh:
Ja, gerne.

Erfahrungsbericht von Kai Meyer

Dieser schriftliche Erfahrungsbericht bezieht sich auf das Teachlet, das Kai Meyer im Rahmen des iPhone-PowerDay am 24. August 2010 an der Universität Hamburg gehalten hat.

Aus meiner Sicht ist das Teachlet sehr gut gelaufen. Das Publikum hat gute Vorschläge gegeben, die in kurzer Zeit zu dem gewünschten Ergebnis geführt haben, zudem gab es mehrere Personen, die sich zu Wort gemeldet haben.

Da es erst am Morgen eine kurze Einführung in die verwendete Programmiersprache gab, habe ich weder nach „Code-Zeilen“ gefragt noch sie erwartet.

Das Feedback aus der Veranstaltung zeigt mir, dass die gestellte Aufgabe verstanden wurde und aufgrund der vorangegangenen Veranstaltungen auch (im Ansatz) gelöst werden konnte.

Während des Teachlets kam es zu keiner Diskussion im Publikum, was zum Teil an der Akustik lag. Eine Diskussion über den Lösungsweg war von meiner Seite aus aber auch nicht geplant. In diesem für die Zuhörer neuen Kontext (Objective-C, iOS, ...) und bei der Anzahl der Zuhörer erschien mir eine (längere) Diskussion nicht sinnvoll.

Die Implementierungsphase wurde schneller als erwartet abgeschlossen. So hatte ich noch ausreichend Zeit, die vorbereitete Version der Anwendung zu demonstrieren. Die zusätzlichen Funktionen rundeten das Teachlet ab.

In einem vergleichbaren Kontext würde ich wieder Teachlets einsetzen, da diese das Verständnis im Publikum verbessert hat.

Nach dem Teachlet und auch während des anschließenden iPhone-Praktikums habe ich von mehreren Personen gehört, dass allein die Aufforderung zu Mitarbeit die Zuhörer mehr motiviert hat, als eine reine Vorlesung. Zudem wurde das Verständnis für die Zusammenhänge im Teachlet verbessert.

Java-Code zum Teachlet ohne Ausgangssystem in BlueJ

```java
public interface BallBeobachter
{
   void ballEreignis(int x, int y, int geschwindigkeit);
}
```

BallBeobachter.java

```java
import java.util.Set;
import java.util.HashSet;
public class Ball
{
   private Set<BallBeobachter> beobachter;

   public Ball()
   {
      this.beobachter = new HashSet<BallBeobachter>();
   }

   public void fuegeBeobachterHinzu(BallBeobachter beobachter)
   {
      this.beobachter.add(beobachter);
   }

   public void entferneBeobachter(BallBeobachter beobachter)
   {
      this.beobachter.remove(beobachter);
   }

   public void ballGetreten()
   {
      int x = (int)(120 * Math.random());
      int y = (int)(90 * Math.random());
      int geschwindigkeit = (int)(140 * Math.random());
      for(BallBeobachter beobachter : this.beobachter)
      {
         beobachter.ballEreignis(x, y, geschwindigkeit);
      }
   }
}
```

Ball.java

```java
public class Kamera implements BallBeobachter
{
   private Ball ball;
   private int id;

   public Kamera(int id)
   {
      this.id = id;
      this.ball = null;
   }

   public void beobachteBall(Ball ball)
   {
      if(this.ball != null)
      {
         this.ball.entferneBeobachter(this);
      }
      this.ball = ball;
      if(this.ball != null)
      {
         this.ball.fuegeBeobachterHinzu(this);
      }
   }

   public void ballEreignis(int x, int y, int geschwindigkeit)
   {
      System.out.println("Richte Kamera " + this.id + " auf (" + x + ";"
        + y + ") aus. Ball fliegt mit " + geschwindigkeit + "km/h");
   }
}
```

Kamera.java

Literatur

[BB07] Beckhaus, Steffi ; Blom, Kristoffer J.: Teaching, Exploring, Learning – Developing Tutorials for In-Class Teaching and Self-Learning. In: *Computer Graphics Forum* 26 (2007), S. 725–736

[Bec99] Beck, Kent: *Extreme Programming Explained: Embrace Change.* Second Edition. Amsterdam : Addison-Wesley Longman, 1999

[Bit06] Bittner, Stefan: *Das Unterrichtsgespräch: Formen und Verfahren des dialogischen Lehrens und Lernens.* Klinkhardt, 2006

[BMM06] Bohnsack, Ralf ; Marotzki, Winfried ; Meuser, Michael: *Hauptbegriffe Qualitativer Sozialforschung.* Stuttgart : UTB, 2006

[CDS09] Cress, Ulrike (Hrsg.) ; Dimitrova, Vania (Hrsg.) ; Specht, Marcus (Hrsg.): *Learning in the Synergy of Multiple Disciplines: 4th European Conference on Technology Enhanced Learning, EC-TEL.* 2009

[CLR04] Cormen, Thomas H. ; Leiserson, Charles E. ; Rivest, Ronald L.: *Algorithmen – Eine Einführung.* Oldenbourg Wissenschaftsverlag, 2004

[FS10] Fink, Marius ; Schomborg, Till: *Teachlet-Dokumentation zum Proxy-Entwurfsmuster.* Ergebnis der Teachlet-Werkstatt, 2010

[HOS97] Heimann, Paul ; Otto, Gunter ; Schulz, Wolfgang: *Unterricht – Analyse und Planung.* Schroedel Verlag GmbH, 1997

[Sch05] Schmolitzky, Axel: A Laboratory for Teaching Object-Oriented Language and Design Concepts with Teachlets. In: *Proc. OOPSLA '05 (Companion: Educators' Symposium).* San Diego, CA : ACM Press, 2005

[Sch07] Schmolitzky, Axel: Patterns for Teaching Software in Classroom. In: *European Conference on Pattern Languages of Programs (EuroPLoP)*, 2007