Vorlesungs-Pflege: Maßnahmen zur Runderneuerung degenerierender Informatikvorlesungen

Karsten Weicker
HTWK Leipzig, Fakultät IMN
Gustav-Freytag-Straße 42A
04277 Leipzig
karsten.weicker@htwk-leipzig.de

Zusammenfassung: Ähnlich zu Alterungsprozessen bei Software degenerieren auch Vorlesungen, wenn sie nicht hinreichend gepflegt werden. Die Gründe hierfür werden ebenso beleuchtet wie mögliche Indikatoren und Maßnahmen – der Blick ist dabei immer der eines Informatikers. An drei Vorlesungen wird erläutert, wie der Degeneration von Lehrveranstaltungen gegengewirkt werden kann. Mangels hinreichend großer empirischer Daten liefert das Paper keine unumstößlichen Wahrheiten. Ein Ziel ist es vielmehr Kollegen, die ähnliche Phänomene beobachten, einen ersten Anker für einen inneren Diskurs zu bieten. Ein langfristiges Ziel ist die Sammlung eines Katalogs an Maßnahmen zur Pflege von Informatikvorlesungen.

Schlüsselworte: Wartung von Lehrveranstaltungen, Re-Engineering, Strukturverbesserung, Lehrevaluation, Degenerationsprozesse.

1 Motivation

Regelmäßig gehaltene Vorlesungen scheinen trotz der zunehmenden Reife, jahrelanger Verbesserung der Beispiele und damit stetig zunehmendem Perfektionsgrad, trotz einer immer ausgefeilteren Vorbereitung und Durchführung der Vorlesung immer weniger erfolgreich zu sein – Studenten[1] zeigen weniger Begeisterung als die Kommilitonen vor zehn Jahren und

1 Alle geschlechterbezogenen Bezeichnungen stehen für Personen jeglichen Geschlechts; also „Studenten“ für „Studentinnen und Studenten“ – analog: Informatiker, Dozent, Kommilitone, Kollege etc.

die Prüfungsergebnisse lassen ebenfalls nach. Ich bezeichne den Effekt als Tao-of-Pooh-Paradoxon nach dem Buchzitat „And when you try too hard, it doesn't work" [Ho82] – trotz gleichbleibender oder gar zunehmender eigener Anstrengungen ist man als Dozent festgefahren in dem Gefüge der eigenen Lehrveranstaltung und kann eher nachteilige Auswirkungen der eigenen Bemühungen beobachten. Wenn sich bei einer wiederholt zu haltenden Vorlesung beim Dozenten das Gefühl einstellt, dass die Dinge eher schlechter als besser werden, bedeutet dies, dass es notwendig sein kann, sich der Vorbereitung anders zu nähern, als man dies bisher gemacht hat. Auch wenn dies kein gesichertes Naturgesetz ist, lässt es sich immer wieder beobachten.

Vielfältige Erklärungsversuche können bemüht werden, um die Ursachen des Tao-of-Pooh-Paradoxon zu beleuchten. Da es in unserem Lehr- und Forschungsgebiet allerdings um Informatikvorlesungen geht, bietet es sich an, das Problem mit den Augen des Informatikers zu untersuchen und entsprechende Lösungsvorschläge aufzuzeigen.

Dieses Paper hat einen Fokus auf der Lehrveranstaltungsform der klassischen Vorlesung – genauer: auf Vorlesungen, die von demselben Dozenten über einen längeren Zeitraum immer wieder gehalten werden. Dabei kann es sich ebenso um Grundlagenveranstaltungen wie um Spezialvorlesungen handeln. Andere Veranstaltungsformen wie Seminare, Praktika, Übungen und Selbstarbeit werden in diesem Paper nicht berücksichtigt, womit die Vorlesung nicht über die anderen Lehrformen gestellt werden soll. Allerdings ist die Vorlesung gerade im technisch-naturwissenschaftlichen Kontext kaum verzichtbar und bedarf deshalb einer besonderen Berücksichtigung.

Abschnitt 2 und Abschnitt 3 stellen frische und degenerierte Vorlesung einander gegenüber. Nach möglichen Indikatoren für eine beginnende Degeneration in Abschnitt 4 zeigt Abschnitt 5 auf, welche Maßnahmen zur Vorlesungspflege zur Verfügung stehen. Einige persönliche Beispiele aus den eigenen Lehrveranstaltungen werden in Abschnitt 6 präsentiert. Die Ausführungen erheben dabei nicht den Anspruch auf eine unumstößliche wissenschaftliche Wahrheit, sondern sie sind streitbar und sollen Kollegen, die ähnliche Phänomene beobachten, einen ersten Anker für einen inneren Diskurs bieten.

2 Frische Vorlesungen

Ein wesentliches Ziel einer Vorlesung ist die Wissensvermittlung. Wie heute hinlänglich bekannt ist, funktioniert dies nicht gemäß des Nürnberger Trichters, sondern es ist notwendig, dass die Studierenden mitdenken und ihr Vorwissen mit den neuen Impulsen verknüpft wird. Allein durch eine aktive Beschäftigung mit den Inhalten können diese Verknüpfungen entstehen.

Um dieses Ziel der aktiven Auseinandersetzung auch in Vorlesungen erreichen zu können, gibt es eine Reihe von didaktischen Maßnahmen wie z. B. aktivierende Methoden. Doch auch darüber hinaus fällt dem Dozenten eine entscheidende Rolle zu, inwieweit es ihm gelingt, den sprichwörtlichen Funken auf die Studenten überspringen zu lassen und ihr Interesse zu wecken. Diese Fähigkeit kann man an den folgenden zwei Eigenschaften festmachen.

- Die *Begeisterung* des Dozenten für seine Inhalte und seine Vorlesung beeinflusst, wie stark sich die Studentinnen und Studenten auf das Fach einlassen – Begeisterung ist ansteckend.
- Die *Vorbildfunktion* des Dozenten als Bild für die Denk- und Herangehensweise von Informatikern. Wird Informatik als reine Routine oder als spannendes, sich ständig weiter entwickelndes Gebiet gesehen und gelebt? Grundsätzlich vermittelt ein Studium in erster Linie Haltungen, Denkweisen und Problemlösekompetenzen. Dies gilt insbesondere und in verstärktem Maß in der Informatik, wo sich Themen, Programmiersprachen und Technologien schnell ändern – laut [HSZ13] bestehen Forschungs-Communities etwa 4,5 Jahre lang, Lehrinhalte wurden 1995 mit einer Halbwertszeit von 5 Jahren beziffert [Sc95]. Vor diesem Hintergrund fällt dem Dozenten eine Vorbildfunktion zu, wie Informatiker in ihrem weiteren Leben mit Neuerungen umgehen sollten: Flexibilität und Aktualität in der Lehre statt eines antiquierten Anstrichs.

Gerade diese beiden Eigenschaften leiden allerdings häufig darunter, wenn dieselben Vorlesungen immer wieder gehalten werden und sich eine Routine breit macht. Im weiteren Text bezeichnen wir eine wiederholt gehaltene Vorlesung als *frisch,* wenn die Begeisterung und die Vorbildfunktion gleichermaßen erhalten bleiben.

3 Degenerierende Vorlesungen

Um die Güte einer Vorlesung zu messen, ist es wichtig, zunächst einmal grundlegende Eigenschaften von Vorlesungen allgemein zu betrachten. Hier drängt sich dem Informatiker ein Vergleich der Eigenschaften von Vorlesungen mit denen von Software auf. Dabei ergeben sich viele z. T. unerwartete Parallelen, aber auch einige Unterschiede. Nach [Ba09] ist Software

- immateriell, was auch für Vorlesungen und vor allem für das zu vermittelnde Wissen selbst gilt (durch ihre inhärente Struktur und den Vortrag selbst von Vorlesungen; einzig die Vorlesungsfolien und ein eventuelles Skript sind in gewissem Sinn „anfassbar"),
- verschleißfrei, was oberflächlich betrachtet auch für Vorlesungen gilt, im Weiteren aber noch etwas näher betrachtet wird, und
- einem Alterungsprozess ausgesetzt, der genauso auch für die Vorlesung gilt (siehe Abbildung 1): Die Umgebung einer Lehrveranstaltung entwickelt sich weiter und verändert sich. Die mathematische Vorbildung von Studienanfängern nimmt ab [Ba13], durch Prägung mittels Messenger-Systemen beträgt deren Aufmerksamkeitsspanne nur noch etwa 280 Zeichen, die Erwartungen, die an Absolventen gestellt werden, ändern sich, wie sich auch die vermittelten Inhalte in Lehrveranstaltungen von Kollegen verschieben und nicht zuletzt entwickelt sich die Informatik weiter und insbesondere (aber nicht ausschließlich) technologielastige Module laufen ständig Gefahr inhaltlich veraltet zu sein. Lehrveranstaltungen, die diesen Änderungen nicht folgen, veralten zwangsläufig.

Zum zweiten Punkt, der Verschleißfreiheit: Eine einmal konzipierte Lehrveranstaltung zu gut gesetzten und gereiften Grundlagen, z. B. Logik, Automaten und formale Sprachen oder Algorithmen und Datenstrukturen, sollte doch problemlos nach 10 oder 15 Jahren noch unverändert funktionieren – wenn man vom oben beschriebenen Alterungsprozess absieht, der sich bei Grundlagenveranstaltungen nur auf veränderte Rahmenbedingungen seitens der Studienanfänger beschränken sollte. Das gilt allerdings nur mit gewissen Einschränkungen, denn das Problem sind wir als Dozenten:

- Erläuterungen zu den Vorlesungsfolien werden über die Jahre hinweg beständig optimiert und leicht verändert – und das trägt nicht zwingend zur Verbesserung der Lehrveranstaltung bei: zunehmende Perfektion verringert die Mitdenklücken für Studierende und manchmal verschieben

sich auch leicht die eigenen Erwartungen, was die Studierenden eigentlich schon wissen sollten.

- Auch die anfängliche Begeisterung des Dozenten für die Vermittlung seiner Inhalte weicht einer Routine, was im Extremfall sogar bis zum Disengagement, also der inneren Kündigung, wie im Schulkontext [SV11] führen könnte. In jedem Fall hat schwindende Begeisterung auf Seiten des Dozenten einen großen Einfluss auf die Wahrnehmung seitens der Studierenden und damit auch auf deren Bereitschaft, sich mit der Materie auseinanderzusetzen.

Zusammenfassend halten wir fest, dass verschiedene Faktoren (vgl. Abbildung 1) dafür sorgen, dass Vorlesungen degenerieren können und damit Maßnahmen der Vorlesungs-Pflege erforderlich sind.

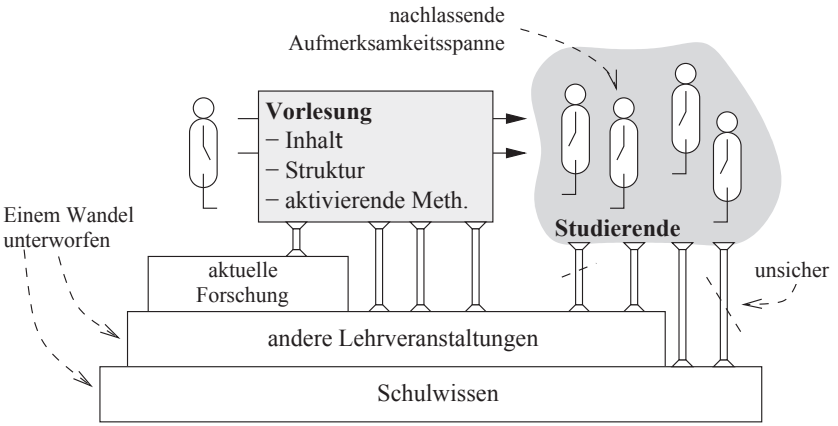

Abb. 1: Überblick über Faktoren, die eine langsame Degeneration einer Vorlesung begünstigen

4 Kriterien für frische Vorlesungen

Wie kann man nun einschätzen, wie frisch eine Vorlesung noch ist? Ideal wäre natürlich eine Metrik als Qualitätskriterium, ähnlich zu Metriken wie wir sie in der Software-Entwicklung kennen. Kandidaten sind

- die Durchschnittsnoten in Klausuren – weniger ein Frischekriterium, sondern von vielen anderen Faktoren überdeckt,
- die Ergebnisse der Lehrevaluation – eher ein Wohlfühlindikator – oder
- das Maß der aktiven Beteiligung in der Vorlesung oder in Übungen – kann sowohl Verständnisschwierigkeiten oder begeistertes Mitdenken bedeuten.

Alle angeführten Qualitätsmerkmale lassen sich zwar messen, ergeben jedoch höchstens in Kombination und unter starker Einbeziehung des Kontexts eine wirkliche Aussage. Alternativ könnte man durch Fragebögen oder andere empirische Methoden über einen längeren Zeitraum hinweg untersuchen, ob sich die studierendenseitige Wahrnehmung der Vorlesung ändert.

Da dieses Paper nicht auf eine empirische Untersuchung abzielt, sondern in erster Linie ein Problembewusstsein wecken soll, bleibt als gut gemeinter Hinweis an Dozenten nur das eigene Gefühl als Indikator und Alarmsignal. Zunehmende Unzufriedenheit mit der studentischen Mitarbeit oder den Ergebnissen in Klausuren und Belegen, zunehmendes Abspulen der Vorlesung oder auch vermehrte kritische Kommentare bei Lehrevaluationen könnten ein erstes Anzeichen für eine degenerierende Vorlesung sein.

5 Maßnahmen zur Vorlesungspflege

Wenn die Analogie zwischen Software und Vorlesungen halbwegs schlüssig ist, bietet sich auch für die Kategorisierung von Maßnahmen der Vorlesungspflege ein Blick auf die Software-Pflege an. Opferkuch und Ludewig [OL04] unterscheiden die folgenden Kategorien:

- Software-Wartung als jede Arbeit an einem Software-System, die unmittelbare Auswirkungen auf den Benutzer hat und nicht von Anfang an geplant war oder hätte geplant werden können – dies möchten wir in unserem Kontext uminterpretieren als jegliche Änderung an einer Vorlesung, die eine inhaltliche Veränderung nach sich zieht, und

- Re-Engineering als absehbare Strukturverbesserung, die der ausschließlichen Verbesserung von Wartungsqualitäten dient – in unserem Kontext jegliche Veränderung an einer Vorlesung bei gleichbleibendem Inhalt (einschließlich Umstrukturierungen, die auch das leichte Verankern von neuen Inhalten ermöglichen).

Beispielhafte Maßnahmen zur Vorlesungswartung sind

- das Bereinigen von Fehlern,
- die Überarbeitung durch eine frische, eigene, in die Tiefe gehende Auseinandersetzung mit dem Inhalt, z. B. durch neue anwendungsnahe oder aktuelle Beispiele, die selbst im Detail aufbereitet werden, bzw. die beispielhafte Umsetzung von Algorithmen in einer aktuellen Programmiersprache,
- die Berücksichtigung von aktuellen Forschungsergebnissen und Veröffentlichungen – dies kann eigene wie auch fremde Forschung sein, aber auch der Bericht zu aktuellen Themen vom Besuch einer Fachtagung – und
- das Infragestellen von Standardlehrstoff innerhalb der Lehrveranstaltung, wie z. B. der Frage warum statt Quicksort der Timsort-Algorithmus in den Java-Collections implementiert ist.

Beim Re-Engineering einer Vorlesung handelt es sich um strukturverändernde Maßnahmen wie z. B.

- jegliches Umstellen der Reihenfolge von Inhalten – dadurch können unterschiedliche Ziele verfolgt werden, wie z. B. die bessere Anknüpfung an Vorwissen, eine bessere Verknüpfung von Inhalten oder durch eine anschließende Vorlesungswartung die Verschiebung des Fokus der Vorlesungseinheit auf ein übergeordnetes Konzept – und
- als Extrembeispiel Konzepte wie Flipped Classroom, die bei gleichbleibenden Inhalten den Ablauf des Vorlesungsalltags maßgeblich verändern und die Rollen zwischen Präsenzveranstaltung und Selbstarbeit tauschen.

6 Beispiele

Ich möchte die Überlegungen an einigen Beispielen aus meinen eigenen Vorlesungen illustrieren.

6.1 Spezialvorlesung „Evolutionäre Algorithmen“

Die Lehrveranstaltung „Evolutionäre Algorithmen“ wird seit 1999 gehalten und hat im Rahmen der Weiterentwicklung des Forschungsgebiets aber auch der Überarbeitung des Lehrbuchs [We15] für Neuauflagen verschiedene Änderungen erfahren. Der grundsätzliche Aufbau der Vorlesung wurde seit 2007 allerdings nur marginal modifiziert.

Daher machten sich Degenerationserscheinungen bemerkbar. Ein anfangs sehr erfolgreicher großer Theorie-/Konzeptblock am Beginn des Moduls funktionierte mit den Jahren immer weniger gut und hat am Ende sogar das Verständnis der Standardalgorithmen erschwert. Dies machte sich auch in den Ergebnissen der Lehrevaluation bemerkbar. So wurde die Frage „Diese Lehrveranstaltung fördert mein Interesse am Thema“ im Sommersemester 2009 mit 1,36 (auf einer Skala von 1 = „sehr gut“ bis 5 = „mangelhaft“) und im Wintersemester 2012/13 mit 1,9 bewertet. Auch bei der Frage „Ich habe in dieser Lehrveranstaltung tiefes Verständnis für den Stoff gewonnen“ verschlechterte sich der Wert von 2,0 auf 2,3.

Eine strukturelle Überarbeitung der Lehrveranstaltung war unumgänglich, die ich in erster Linie dazu genutzt habe, Konzepte und Standardalgorithmen parallel einzuführen und so klarer die Bezüge dazwischen zu verdeutlichen. Abbildung 2 stellt bildlich dar, wie die Inhalte aus den bisherigen zwei großen Abschnitten zu Konzepten und Standards umverteilt und wie ein Sandwichbelag in kleinere Einheiten gebettet werden. Dasselbe Prinzip wurde auch mit den Fallstudien am Ende der Vorlesung angewandt, wo jeweils eine Fallstudie mit Techniken zum Umgang mit speziellen Anforderungen verknüpft wird.

Das reine Re-Engineering der Vorlesung genügte mir an dieser Stelle allerdings nicht, sondern ich entschloss mich zusätzlich zu den bewährten Algorithmenvisualisierungen kleine Programmbeispiele mit einzuflechten. Hierbei war das Ziel, einerseits die einfache Umsetzung der vorgestellten Ideen zu betonen und andererseits mehr Spaß und Begeisterung in die Vorlesung zu bringen. Daher habe ich die Beispiele in der noch relativ jungen Sprache Go implementiert. In jeder der ersten sieben Vorlesungseinheiten wird ein

kleines Beispielprogramm präsentiert, dessen Quelltext genauer analysiert und damit kleine Optimierungsexperimente durchgeführt.

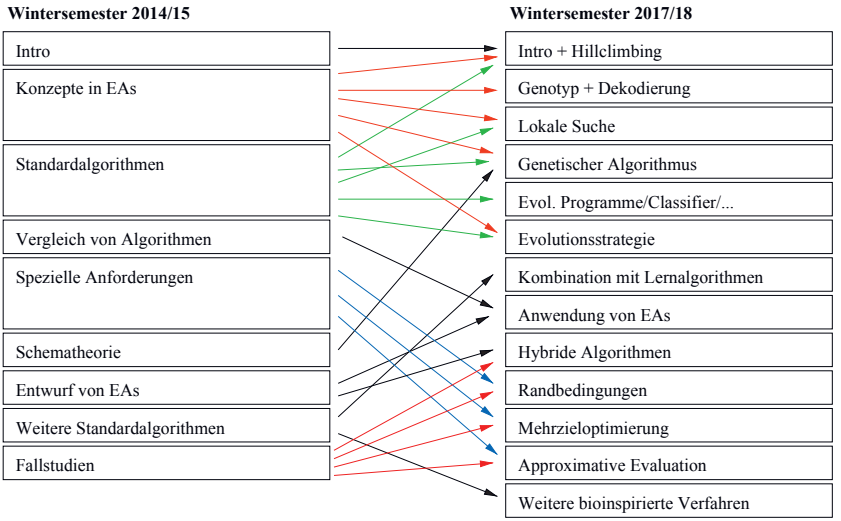

Abb. 2: Vergleich der Struktur des Moduls „Evolutionäre Algorithmen“ vor und nach der Restrukturierung

Die Lehrveranstaltung wurde erstmalig in dieser Form im Wintersemester 2017/18 gehalten und auch wieder durch die Lehrevaluation begleitet. Die Frage „Diese Lehrveranstaltung fördert mein Interesse am Thema“ wurde jetzt mit 1,3 und die Frage „Ich habe in dieser Lehrveranstaltung tiefes Verständnis für den Stoff gewonnen“ mit 1,8 bewertet. In den Freitextkommentaren wurde insbesondere die Verwendung von Go bzw. von vielen praktischen Demonstrationen und Beispielen positiv hervorgehoben, wobei die typischen Gefahren von viel Quelltext in einer Vorlesung auch hier negativ vermerkt wurden und das Vorstellen von Code von einem Zuhörer als „oft verwirrend und nicht hilfreich beim Verstehen“ bezeichnet wurde. Insgesamt hat die Lehrveranstaltung durch die Veränderungen neuen Schwung bekommen, was meines Erachtens nach maßgeblich an der neu entfachten Begeisterung des Dozenten für die frische (und manchmal auch noch nicht ganz perfekt ausgearbeitete) Vermittlung des Lehrinhalts liegt.

6.2 Grundlagenveranstaltung „Algorithmen und Datenstrukturen“

An der HTWK Leipzig findet die Lehrveranstaltung „Algorithmen und Datenstrukturen“ im zweiten Fachsemester des Bachelorstudiums statt. Auch diese Lehrveranstaltung wurde mehreren Strukturveränderungen unterworfen. Die größte Modifikation war im Jahr 2007 die Neustrukturierung gemäß eines Spiralkonzepts, das bereits an anderer Stelle [We07] beschrieben wurde: fünf algorithmische Probleme dienen als roter Faden, der sich durch die Lehrveranstaltung zieht und an dem verschiedene algorithmische Ideen verankert werden (vgl. linker Teil von Abbildung 3). Der Aufbau führte zu einer wesentlich verbesserten Vermittlung der involvierten Konzepte.

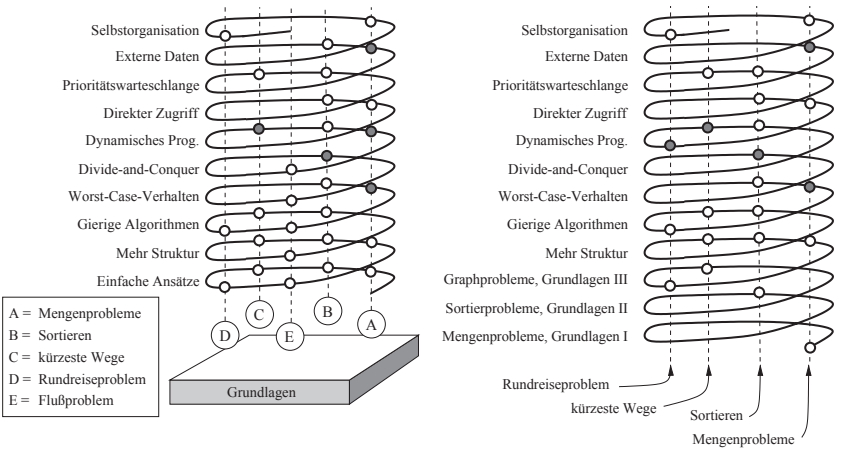

Abb. 3: Vergleich der Struktur der Vorlesung „Algorithmen und Datenstrukturen“. Links das ursprüngliche Spiralkonzept, rechts die Überarbeitung mit dem veränderten Einstieg in die Lehrveranstaltung

Allerdings hinterließen neun Jahre bis zum Jahr 2016 auch bei dieser Lehrveranstaltung Spuren der Degeneration. Maßgeblich haben sich in dieser Zeit die Vorkenntnisse der Studierenden verändert, da sich Ausrichtung und Inhalt von Lehrveranstaltungen im ersten Fachsemester leicht verschoben haben.

Die konkreten Maßnahmen waren hier weniger einschneidend. Ähnlich zum Re-Engineering der Spezialvorlesung in Abschnitt 6.1 wurden auch hier die Grundlagen mit den einfachen Lösungsansätzen verschränkt, sodass eine

stärker im Beispielproblem verhaftete Einführung der Grundlagen entstand (vgl. Abbildung 3). Unabhängig von der Re-Strukturierung wurde durch Verkürzung der Vorlesungszeit an der Hochschule die Menge der betracheten algorithmischen Probleme um ein Problem verkleinert.

6.3 Praxisorientierte Lehrveranstaltung „Softwaretechnik“

Das Modul „Softwaretechnik“ wird von mir seit 2004 als Lehrveranstaltung des dritten Bachelorfachsemesters gehalten und zeigt insgesamt wesentlich weniger Symptome der Degeneration. Das liegt daran, dass die Lehrveranstaltung durch Überarbeitungen mindestens alle zwei Jahre einem großen empirischen Dauerexperiment gleicht, bei dem ich regelmäßig Inhalte erneuere oder austausche. Die Auslöser hierfür sind vielfacher Natur: der Wegfall einer Folgeveranstaltung „Softwaretechnik II“, aktuelle Entwicklungen in der Branche oder die eigene Verschiebung von Schwerpunkten. Zwei wesentliche Einflussfaktoren waren die Einführung von Programmierübungen im Computerpool sowie die parallel startende Lehrveranstaltung des Softwareprojekts.

Die Übungen im Poolraum finden 14-tägig statt und es müssen im Rahmen des 90-minütigen Termins verschiedene Aufgaben individuell gelöst werden. Die Themenblöcke umfassen Source-Code-Management, Testen mit JUnit, Entwurfsmuster, Refactoring, Modellierung mit UML und werkzeuggestützte Code-Erzeugung. Der Zwang, über den gesamten Verlauf der Lehrveranstaltung beständig auch praktisch ein- und umsetzbare Fachinhalte zu präsentieren, hatte einen positiven Effekt auf den Erhalt der Frische der Lehrveranstaltung.

Ähnlich positiv hat sich das Softwareprojekt ausgewirkt, das in größeren Teams mit Masterstudenten als Projektleiter parallel im dritten Semester startet und über den Zeitraum von zwei Semestern läuft. Als Konsequenz muss die Lehrveranstaltung „Softwaretechnik“ beständig und rechtzeitig Prozesswissen für die Durchführung von Softwareprojekten liefern, welches durch die Bachelorstudenten auch fortwährend mit ihren praktischen Erfahrungen im größeren Projektkontext abgeglichen wird.

7 Diskussion und Ausblick

Erfolgreiche Lehrveranstaltungen sind kein Selbstläufer sondern müssen gezielt gepflegt werden. Hierbei hilft zumindest dem Informatikdozenten die Analogie zur Software-Entwicklung. Als wesentliches Ziel sollte dabei die Begeisterung für den Stoff und die Veranstaltung sowohl beim Lehrenden als auch bei den Studierenden erhalten werden. Die Beispiele in diesem Paper stützen sich stark auf den Re-Engineering-Aspekt, also der strukturellen Überarbeitung eines Moduls. Hierfür können didaktische Konzepte oder auch einfache pragmatische Gedanken als Leitidee der Überarbeitung wirksam werden. Drei typische Muster kamen in den präsentierten Beispielen zur Anwendung: das Spiralkonzept als iterierendes, Wissen in Schichten aufbauendes, didaktisches Prinzip, die sandwichartige Verschränkung von Grundlagen bzw. Konzepten mit Standardalgorithmen und die Einbettung von Fallbeispielen in den fortgeschrittenen Teil der Lehrveranstaltung.

Für die Zukunft wäre es wünschenswert, wenn ein konkreter Katalog von möglichen Maßnahmen für die Pflege von Vorlesungen gesammelt werden könnte – ähnlich zu den Entwurfsmustern bei der Software-Architektur, die auch für typische Probleme durch ein Schema eine Lösung anbieten. Auch hierfür bietet es sich an, den Kontext noch weiter zu öffnen – wie das Beispiel des Moduls „Softwaretechnik“ zeigt, ist das Potential der möglichen Ansätze zur Pflege einer Lehrveranstaltung wesentlich größer, wenn man betreute Übungen im Computerpool, Projektarbeiten oder die Schnittstellen zu parallel gehaltenen Lehrveranstaltung berücksichtigt. Auch aus der Hochschuldidaktik können noch wesentlich mehr Impulse berücksichtigt werden, man denke nur an problembasiertes Lernen oder Learning-by-Teaching.

Literaturverzeichnis

[Ba09] Balzert, H.: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. Spektrum Akademischer Verlag, 2009, S. 9.

[Ba13] Baumann, A.: Mathe-Lücken und Mathe Legenden: Einige Bemerkungen zu den mathematischen Fähigkeiten von Studienanfängern. Die Neue Hochschule, 2013(5), S. 150–153, 2013.

[Ho82] Hoff, B.: The Tao of Pooh, E. P. Dutton, 1982.

[HSZ13] Hoonlor, A.; Szymanski, B. K.; Zaki, M. J.: Trends in Computer Science Research. Communications of the ACM, 56(10), S. 74–83, 2013.

[OL04] Opferkuch, S.; Ludewig, J.: Software-Wartung – Eine Taxonomie. Softwaretechnik-Trends, 24(2), S. 35–36, 2004.

[Sc95] Schinzel, B.: „Welchen Wert haben theoretische Grundlagen in der Berufspraxis? Was Theorie leisten kann und soll.“ Universalismus, Abstraktion und Modellbildung in der Informatik. In (Huber-Wäschle, F.; Schauer, H.; Widmayer, P. Hrsg.): GISI 95: Herausforderungen eines globalen Informationsverbundes für die Informatik, Springer, S. 366–373, 1995.

[SV11] Schmitz, E.; Voreck, P.: Einsatz und Rückzug an Schulen: Engagement und Disengagement bei Lehrern, Schulleitern und Schülern, VS Verlag, 2011.

[We07] Weicker, N.: Zielorientierte Didaktik der Informatik – Kompetenzvermittlung bei engen Zeitvorgaben. In (Schubert, S. E. Hrsg.): Didaktik der Informatik in Theorie und Praxis. INFOS2007, Lecture Notes in Informatik 207, Gesellschaft für Informatik, S. 337–348, 2007.

[We15] Weicker, K.: Evolutionäre Algorithmen, 3. Auflage, Springer Vieweg, 2015.