Planspiel und Briefmethode für die Software Engineering Ausbildung - ein Erfahrungsbericht

Georg Hagel, Hochschule Kempten

georg.hagel@fh-kempten.de

Jürgen Mottok, LaS³, Hochschule Regensburg

juergen.mottok@hs-regensburg.de

Zusammenfassung

Die konstruktivistische Perspektive unterstützt einen Paradigmenwechsel der akademischen Lehre hin zu einer Lerner- und Lernprozesszentrierung. Dabei thematisieren Selbstgesteuertes Lernen und aktivierende Lehre das studentische Lernen neu.

Das Zusammenspiel von Lehren und Lernen mit dem Ansatz einer konstruktivistischen Didaktik wird mit Beispielen erprobter Lernkonzepte der Software Engineering Ausbildung der Vortragenden unterlegt: Planspiel und Briefmethode werden exemplarisch diskutiert.

Die gezeigten konstruktivistischen Methoden lassen sich auf Lebenslanges Lernen des Software Engineering im Berufsumfeld übertragen.

Einführung – Lernarrangements für selbstgesteuertes Lernen

Inzwischen ist eine Wende der didaktischen Wahrnehmung erkennbar. Während die curriculumtheoretische Didaktik auf der Überzeugung basierte, dass Lernprozesse Erwachsener zielgerichtet planbar und steuerbar sind, zielt der konstruktivistisch-didaktische Ansatz auf die Ausgestaltung von Lernumgebungen (Siebert, 2009). In diesen Lernumgebungen, auch Lernsettings, genannt, sollen selbstgesteuerte und kreative Lernprozesse angeregt werden. Dabei ist nicht nur das Expertenwissen der Referenten eine Ressource, sondern auch das Vorwissen, die Erfahrungen und die Fragestellungen aller Beteiligten. Dieser methodisch didaktische Ansatz gestaltet einem Paradigmenwechsel „The Shift from Teaching to Learning“ (Welbers, 2005) aus.

Konstruktivismus

Der Konstruktivismus ist eine Theorie über den Erwerb von Wissen, das Lernen und Lehren. Kernaussage ist, dass jeder Mensch durch die Kommunikation mit seiner Umgebung eine eigene persönliche Wirklichkeit erschafft; diese unterscheidet sich von der Wirklichkeit anderer Menschen. Lernen wird als die Konstruktion von Bedeutung und damit als dynamisches Weiterentwickeln der persönlichen Wirklichkeit gesehen. Lernen im didaktisch konstruktivistischen Kontext unterscheidet:

1. Konstruktion („Wir sind Erfinder unserer Wirklichkeit“),
2. Rekonstruktion („Wir sind die Entdecker unserer Wirklichkeit“) und
3. Dekonstruktion („Es könnte auch anders sein! Wir sind die Enttarner unserer Wirklichkeit!“).

Die grundsätzliche Ausrichtung ist: „Selbst erfahren, ausprobieren, untersuchen, experimentieren, immer in eigene Konstruktion ideeller oder materieller Art überführen und in den Bedeutungen für die individuelle Interessen-, Motivations- und Gefühlslage thematisieren.“ (Reich, 2008)

In der Perspektive der Rekonstruktion lautet die Frage: „Wer hat es damals so und wer hat es anders gesehen? Welche Handlungsmöglichkeiten haben Beobachter damals festgestellt und welche fallen uns hierzu ein? Welche unterschiedlichen Experten kommen zu welcher Aussage und wie stehen wir dazu?“ In dieser Perspektive wird gefragt, welche Motive der damalige Beobachter hatte um seine Festlegungen zu treffen. Faktenwissen steht dabei nicht im Vordergrund.

Die Dekonstruktion stellt sich die Frage der selbst vollzogenen Auslassungen, die möglichen anderen

Blickwinkel, die sich im Nachentdecken der Erfindungen anderer oder in der Selbstgefälligkeit der eigenen Erfindung so gerne einstellen. In dieser Perspektive will der Enttarner kritisch gegenüber den eigenen blinden Flecken sein.

Als idealtypischer Grundsatz für die konstruktivistische Didaktik gilt somit (Reich, 2008):

„Jeder Sinn, den ich selbst für mich einsehe, jede Regel, die ich aus Einsicht selbst aufgestellt habe, treibt mich mehr an, überzeugt mich stärker und motiviert mich höher, als von außen gesetzter Sinn, den ich nicht oder kaum durchschaue und der nur durch Autorität oder Nicht-Hinterfragen oder äußerlich bleibende Belohnungssysteme gesetzt ist."

Der konstruktivistische Methodenbaukasten in der Software Engineering Ausbildung

Die Software Engineering Studenten müssen von Anfang an nicht nur in Erkenntnistheorie, sondern auch in Problemlösung vertraut werden (Ludewig, 2009). Dies wird durch den Perspektivwechsel von der Input- zur Output-Orientierung unterstützt, wobei durch den Einsatz geeigneter fachdidaktischer Methoden die Lernenden ihren Lernprozess in Selbststeuerung aktiv ausgestalten.

Den Lehrenden stehen mit den konstruktivistischen Methodenbaukästen „Methodenpool" (Reich, 2008) und der „Methodensammlung" (Macke, 2009) Ideenquellen zur Ausgestaltung eines aktivierenden Lernprozesses zur Verfügung. Erste positive Versuche im Einsatz dieser Methoden findet man beispielsweise in (Mottok, 2009 und Hagel, 2010). Im Folgenden werden die Erfahrungen mit den Methoden Planspiel und Briefmethode im Fach Software Engineering vorgestellt.

Planspiel

In Planspielen im Fach Software Engineering sollen Studierende durch Simulation einer Praxissituation einen möglichst realistischen und praxisbezogenen Einblick in Probleme und Zusammenhänge der methodischen Softwareentwicklung gewinnen, eigene Entscheidungen treffen und Konsequenzen ihres Handelns erfahren.

Planspiele erfordern zudem eine hohe Partizipation aller Beteiligten. Sie sollten auf eine Erhöhung der Handlungsfähigkeit in dem Sinne zielen, dass sie Konsens und Dissens, Entscheidungsabläufe und Transparenz bei der Bildung von Gruppenentscheidungen aufdecken und diskutierbar werden lassen (Reich, 2008). Die Studierenden werden im Planspiel mittels aktivierender Methoden beteiligt und in ein Software-Projekt involviert. Die Aufgabenstellung eines Softwareprojekts und die einzunehmenden Rollen sind vorgegeben. Das Ergebnis des Planspiels bleibt insofern offen, als dass die Lernenden verschiedene Lösungswege auffinden können.

Im Curriculum des Bachelorstudiengangs Mechatronik der Hochschule Regensburg wird im vierten Semester die Vorlesung Software Engineering im Umfang von 2 SWS/3 ECTS und im fünften Semester das Praktikum/Seminar als Blockveranstaltung Software Engineering im Umfang von 4 SWS/5 ECTS angeboten. Das Praktikum/Seminar Software Engineering verlangt also eine Arbeitszeit von 150 Stunden.

Die Blockveranstaltung Software Engineering (50 Stunden) hat zwei vorgelagerte Vorlesungstermine (je 4 Stunden) zur Vorbesprechung. Danach beginnt schon eine selbstgesteuerte Arbeitsphase der Studierenden. Als Vor- und Nachbereitungszeit der Studierenden verbleiben damit 92 Stunden.

Alle Studierenden haben als Vorkenntnisse die Programmiersprachen C und C++ (10 SWS V+Ü), sowie Mikrokontrollertechnik (6 SWS V+Ü). Sie haben dagegen keine Erfahrung über die bei der Programmierung hinausgehenden Schritte der Software-Entwicklung. Praktika und Projekte sind deshalb für die Lehre von Software Engineering zentral (Ludewig, 2009).

Die Vorlesung Software Engineering bereitet auf eine schriftliche Prüfung vor. Dagegen wird die Blockveranstaltung Praktikum/Seminar Software Engineering mit einem studienbegleitenden Leistungsnachweis abgeschlossen. Der studienbegleitende Leistungsnachweis wird erbracht durch die Vorbereitung eines Fachvortrages, die Vorbereitung eines Posters, die erfolgreiche Mitwirkung am Planspiel und die Erstellung von Arbeitsprodukten, wie qualitätsgesicherter Dokumentationen, sowie funktionierender Software.

Die Blockveranstaltung Praktikum/Seminar Software Engineering an der Hochschule Regensburg findet an fünf Tagen statt. Zwei Lehrende gestalten mit Pair-Teaching als interdisziplinäres Team (Informatiker und Pädagoge/Projekttrainer) diese Veranstaltung. Insbesondere diese Verzahnung im Führungstandem lässt für die Studierenden das Wechselspiel zwischen Konkurrenz versus Zusammenarbeit/Dialog in einem Klima offener Kommunikation sichtbar werden. Eine Übertragung auf die eigene Situation im Planspiel wird möglich.

Während in den ersten beiden Tagen Fachthemen erarbeitet und vermittelt werden, werden in den letzten drei Tagen in einem Planspiel die Kenntnisse und Fertigkeiten in den Fachthemen vertieft und angewendet.

Bereits vier Monate vor der Blockveranstaltung finden zwei Vorbesprechungen im Umfang von jeweils 4 Stunden mit den Studierenden statt. Dabei

werden Fachthemen zur Vorbereitung der Blockveranstaltung an die Studierenden vergeben. Diese Fachthemen werden von den Studierenden eigenständig bearbeitet. Als Ergebnis werden die Fachvorträge als Folienpräsentation und Poster in die Lernplattform moodle abgelegt. Der Lehrende gibt zu diesen Ergebnissen Rückmeldung in moodle.

Den Studierenden werden zusätzlich Literaturhinweise zur Bearbeitung der Aufgaben in der Lernplattform moodle zur Verfügung gestellt. Die Lernplattform ist vor und während des Planspiels im Einsatz.

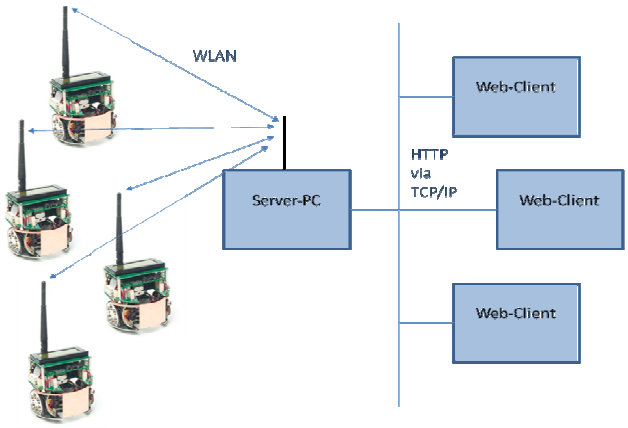


Abbildung 1: Software-Architektur mit c't-Bot, Server-PC und Web-Clienten für das Planspiel

Das durchgeführte Planspiel im Fach Software Engineering behandelt eine Projektaufgabe mit dem Embedded Roboter System c't-Bot. In dieser Aufgabe soll eine Fernsteuerung- und Fernüberwachung des Roboters c't-Bot über einen Server-PC und zusätzlich über Web-Clienten, also Browserapplikationen, erstellt werden (Abbildung 1). Bibliotheken und einfache Beispiele liegen bereits vor.

Zur Durchführung eines Planspiels im Software Engineering müssen folgende Spielmaterialien bereitgestellt werden:

- Eine Fallstudie, in der kurz die vorgegebene Softwareaufgabe skizziert wird und Software-Bibliotheken, sowie bestehende exemplarische Teillösungen vorgegeben werden.
- Eine Arbeitskarte mit Erläuterungen zum Verlauf des Softwareprojekts (Spielverlauf).
- Rollenkarten, durch welche den Teilnehmern spezifische Rollen übertragen werden (Die Softwareentwicklungsprozesse V-Modell 97 und V-Modell XT wurden bereits in der Vorlesung angesprochen.). Die Studierenden nehmen damit die Positionen einer Rolle (Projektleiter, Software Entwickler, Qualitätsmanager, Konfigurationsmanager, …) an.
- Ereigniskarten, die als Impulskarten durch den Spielleiter in die Gruppen gereicht werden können (beispielsweise die Änderungen von Anforderungen).
- Quellen und Literatur

Die einzelnen Phasen des Planspiels bestehen aus:

1. Spieleinführung

Die Semestergruppe der Studierenden wird zu Beginn des Planspiels in mehrere Planspielgruppen mit jeweils ca. 10 Studierenden aufgeteilt. Der Lehrende gibt die Ausgangslage schriftlich vor und klärt Verständnisfragen. Das Spielmaterial wird vorgestellt.

2. Informations- und Lesephase, Rollenverteilung

Die Gruppen erhalten die Rollen- und Arbeitskarten. Das Arbeitsmaterial wird durchgelesen und auftretende Verständnisfragen werden geklärt. Die Teambildung und Rollenverteilung wird von den Lehrenden begleitet.

3. Meinungsbildung und Strategieplanung innerhalb der Gruppe

Die Informationen werden gruppenintern strukturiert und die Projektaufgabe der Softwareentwicklung wird analysiert.

4. Interaktion zwischen den Rollen

In dieser intensivsten Spielphase agieren die Rollen der jeweiligen Planspielgruppe miteinander. Interessenkonflikte zwischen den Rollen treten auf. Diese Interessengegensätze sind typisch bei der Durchführung eines Planspiels. Durch Ereigniskarten kann der Spielleiter nun gezielte Impulse und Veränderungen ins Spiel bringen. Alle Planspielgruppen treffen sich zweimal am Tag zu einem Jour Fix. Die Rolleninhaber müssen dabei unter Zeitdruck Entscheidungen treffen.

5. Vorbereitung eines Plenums / Konferenz

Jeder Rollenträger/Positionsinhaber der jeweiligen Gruppe trägt intern seine Ergebnisse zusammen und verarbeitet und bewertet in dieser Phase die erreichten Ergebnisse.

6. Durchführung eines Plenums / Konferenz

Die Ergebnisse des Software-Projektes werden aus der Perspektive des jeweiligen Rollenträgers vorgestellt. Eine Demonstration mit dem realen software-intensiven System eines Roboters ist gewünscht.

7. Spielauswertung

Auswertung des Spielverlaufs mit dem Lehrenden als neutralen Moderators. Diese Reflexion über den

eigenen Lernprozess ist ein weiteres Merkmal eines Planspiels.

Während der Blockveranstaltung Praktikum/ Semnar Software Engineering stehen ausreichend Semnar- und Rechnerräume für die einzelnen Planspielgruppen zur Verfügung.

Der Ergebnispräsentation am Ende des Planspiels folgt eine Reflexion über den Lernprozess.

In der Reflexion wurden folgende Erfahrungen gesammelt und evaluiert:

- Die Lernthemen können von den Studierenden mitbestimmt werden (Rolleninhaber bereiten Themen vor)
- Der Lehrende übt als Spielleiter keine dominante Rolle aus, sondern ist Begleiter des Lernprozesses und berät bei Rückfragen (Aviram, 2000).
- Offene Form des Lernens ermöglicht die einzelnen Aufgaben zu differenzieren und zu individualisieren (Macke, 2009).
- Qualitätssicherung durch Literaturvorgaben, sowie Begleitung und Rückkopplung mit der Lernplattform moodle, - auch schon vor der Blockveranstaltung.
- Eigene Lernprojekte konnten aus der Vorbereitung der Fachthemen eingebracht werden.
- Die Lernorganisation des Planspiels lässt mehrere Lernwege offen, - Anknüpfung an die Lebens- und an die Praktikumserfahrung der Lernenden.
- Förderung der Handhabung verschiedenster Arbeitstechniken.
- Die Lerninhalte sind mit dem Anwendungsfall der Projektarbeit fassbar reduziert.
- Die angebotenen Lerninhalte können selbsttätig erschlossen werden.
- Handlungsbezogene Problemstellungen im Planspiel sind explizit Thema in der Blockveranstaltung.
- Bereichsübergreifendes Denken und Handeln wird gefördert, ebenso wie ein Verständnis für gruppendynamische Prozesse und ihre Auswirkungen.
- Komplexe Themen, wie Projektmanagement, Qualitätssicherung und Konfigurationsmanagement können in der zur Verfügung stehenden Zeit nur in grundlegender Weise vermittelt werden. Eine Vertiefung kann für Mechatronik-Studierende erst im Masterangebot erfolgen.
- Jede Planspielgruppe entwickelt eine andere Kultur.
- Lernen in multiplen Kontexten
- Mit Verlassen des 90-Minutenrythums entsteht Raum, Zeit und Gelassenheit zum Lernen.

Das Planspiel beinhaltet eine große Menge anderer Methoden und Techniken (Methodeninterdependenz), in denen sich der Studierende üben kann. Im Einzelnen sind dies die Arbeitsform der Gruppenarbeit, Strukturierung der Gruppenarbeit durch Moderation, Ideenentwicklung durch Clustering und Concept Learning, sowie Feedback zur Klärung von Gruppenkonflikten.

Der Unterschied zu Lernformen wie der Projektarbeit besteht darin, dass es noch stärker die Entwicklung von Handlungs- und Entscheidungskompetenzen und das Einüben entsprechender Verhaltensweisen betont (Markowitsch, 2004). Die Begründung von Architekturentscheidungen, die Auswahl möglicher Alternativen in Design und Implementierung, die Festlegung eines Testkonzeptes, aber auch die Ausgestaltung qualitätssichernder Review-Sitzungen sind als Beispiele zu nennen. Diese Beispiele finden sich zwar auch in Projektarbeiten wieder, aber im Planspiel wird die soziale Interaktion der Rolleninhaber und die gemeinsame Reflexion über den Lernprozess am Spielende als methodisches Merkmal genannt. Insofern kann die dargestellte Lernform als projektorientiertes Planspiel klassifiziert werden.

An der Blockveranstaltung Software Engineering haben bereits Semestergruppen mit 20 bis 60 Studierenden teilgenommen.

Mithilfe eines standardisierten Fragebogens konnten Werte für die Zufriedenheit der Studierenden mit der Blockveranstaltung Software Engineering ermittelt werden. Insbesondere wurden Fragen zur Veranstaltung selbst, den technischen Lerneinheiten und zur Begleitung durch den Lehrenden gestellt. Bei der Zufriedenheit handelt es sich um Einschätzungen der Beteiligten selbst, d.h. die relativen Werte für die Zufriedenheit sind sehr repräsentativ und spiegeln die Stimmungen angemessen wieder. An der Umfrage nahmen 80% aller Studierenden teil, sodass die Ergebnisse für den ganzen Kurs geltend gemacht werden können. Die Evaluationsergebnisse waren durchweg positiv.

Kritisch zu bewerten ist, dass die Studien- und Prüfungsordnung für Bachelorstudierende der Mechatronik insgesamt nur 2 SWS Vorlesung und 4 SWS Praktikum/Seminar für das Lehrgebiet Software Engineering vorsieht. Deshalb können die Studierenden sich nur grundlegende Kenntnisse in der disziplinierten Software-Entwicklung bei Anwendung von Softwareprozessmodellen aneignen.

Briefmethode

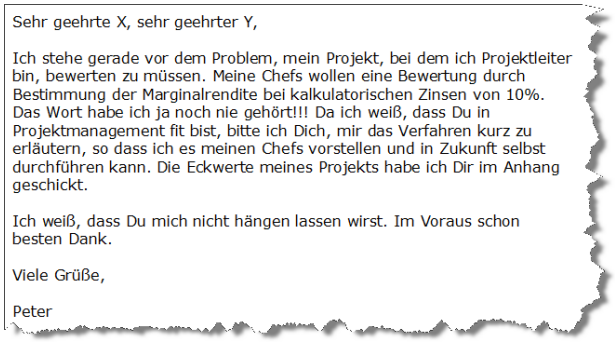
Auch die Briefmethode stammt aus dem Konstruktivistischen Methodenpool (Reich, 2008). Diese Me-

thode wird häufig im Deutschunterricht der Schulen, aber auch in Geschichte und Literatur eingesetzt. Wir wollten untersuchen, ob sich diese Methode auch für den Einsatz in einem technischen Fach, wie Software Engineering an einer Hochschule eignet. Dabei wurden seitens der Dozierenden mehrere Ziele verfolgt: Die Studierenden sollten

- einen Sachverhalt, den sie sich vorher erarbeitet hatten, wiedergeben können und
- lernen, Briefe mit technischer Information zu verfassen.

Das Erstellen technischer Dokumentation kommt aus Sicht der Autoren in den Lehrveranstaltungen zum Software Engineering zu kurz und beschränkt sich meist auf die Erstellung von UML-Diagrammen. Texte mit technischem Inhalt, oder gar Benutzerdokumentation wird sehr selten im Rahmen der Software Engineering-Ausbildung in den Hochschulen verfasst. Ein Versuch, wie eine Ausbildung im Technischen Schreiben aussehen kann, findet man in (Schmidt, 2009). Auch werden an verschiedenen Hochschulen inzwischen explizit Veranstaltungen wie „Software Engineering und Technisches Schreiben“ angeboten.

In der Vorbereitung muss zunächst ein hinreichend komplexer Sachverhalt gefunden werden, der sich in Briefform gut vermitteln lässt. In unserem Beispiel bat der Dozierende die Studierenden um ein Antwortschreiben auf einen fiktiven Brief (siehe Abbildung 2). Im Brief bittet ein Freund der Studierenden um Hilfe bei einer betriebswirtschaftlich-mathematischen Aufgabe aus dem Projektmanagement.


Sehr geehrte X, sehr geehrter Y,

Ich stehe gerade vor dem Problem, mein Projekt, bei dem ich Projektleiter bin, bewerten zu müssen. Meine Chefs wollen eine Bewertung durch Bestimmung der Marginalrendite bei kalkulatorischen Zinsen von 10%. Das Wort habe ich ja noch nie gehört!!! Da ich weiß, dass Du in Projektmanagement fit bist, bitte ich Dich, mir das Verfahren kurz zu erläutern, so dass ich es meinen Chefs vorstellen und in Zukunft selbst durchführen kann. Die Eckwerte meines Projekts habe ich Dir im Anhang geschickt.

Ich weiß, dass Du mich nicht hängen lassen wirst. Im Voraus schon besten Dank.

Viele Grüße,

Peter

Abbildung 2: Brief

Die Studierenden, die eine Marginalrenditerechnung schon mehrmals in Übungen gelöst hatten, sollten "ihrem Freund" in Briefform antworten, also durch selbstständiges Handeln ein neues Produkt, nämlich die technische Lösung der Aufgabe als Brief erstellen.

Ergebnis war, dass lediglich 20% der Studierenden einen Antwortbrief verfasst hatten. Auf Nachfrage seitens des Dozierenden stellte sich heraus, dass viele mit einer so gestellten Aufgabe nichts anfangen konnten. Daher ist es speziell in technischen Fächern sinnvoll, den Einsatz der Methode zu motivieren. Ist den Studierenden der Sinn dieser Art Aufgabenstellung transparent, erhöht sich der Rücklauf beträchtlich.

Diejenigen Studierenden, die das Antwortschreiben verfasst hatten, haben sehr gut verständliche technische Dokumente abgeliefert. Dabei wählten sie selbstständig und unabhängig voneinander unterschiedliche Formate für das Antwortschreiben. Die einen antworteten mit einem kommentierten Excel-Sheet, die anderen mit Word-Dokumenten. Der Dozierende hat auf die Antwortbriefe wieder per Brief individuell Feedback gegeben.

In der Reflexion der Methode mit den Studierenden stellte sich heraus, dass sie das Verfassen des Briefes schwierig fanden: Sie mussten

- einen technischen Text sauber formulieren und übersichtlich strukturieren,
- erkennen, wo sie noch Lücken hatten, sich die Terminologie nochmals aneignen,
- ohne direkte Kommunikation einen Sachverhalt schildern und
- Empathie entwickeln für jemanden, der die gestellte Aufgabe nicht eigenständig lösen kann.

Positive Rückmeldungen seitens der Studierenden waren

- Sie verstehen das Thema jetzt, wo sie es jemand anderem erklären mussten wesentlich besser.
- Sie fanden das konstruktive Feedback des Dozierenden sehr hilfreich.

Das durchweg positive Feedback der Studierenden, die diese Aufgabe gelöst hatten ermutigt den Dozierenden, diese Methode zukünftig häufiger einzusetzen, damit die Studierenden mit dieser Art der Aufgabenstellung vertraut werden. Damit wird der Rücklauf bei der Methode sicherlich erhöht.

Eine spezielle Ausprägung der Briefmethode ist ein Online-Forum, das die Studierenden in Eigeninitiative eingerichtet haben. Dort sind Studierende **und** Dozierende angemeldet. Studierende können hier Fragen, Anregungen oder Kritik jederzeit äußern. Speziell zu technischen Problemen kommen häufig Fragen im Forum und werden oft von den Studierenden selbst beantwortet, so dass diese einiges an Erfahrung in technischer Dokumentation aufbauen. Allerdings ist der Schreibstil im Forum nicht immer technisch und formal korrekt, was auch nicht beabsichtigt ist. Auch Anrede und Grußformel fehlen. Allerdings kann durch den Einsatz von Emoticons eine Aussage unterstrichen und die Kommunikation aufgelockert werden. Die-

se Möglichkeit der Hilfe zur Selbsthilfe wird auch seitens der Dozierenden sehr begrüßt und unterstützt: Auf Fragen wird geantwortet und Kritik und Anregungen zukünftig berücksichtigt.

Damit das Forum funktioniert und regelmäßig benutzt wird, ist es notwendig, dass die Dozierenden regelmäßig und zeitnah antworten. Es wird seitens der Studierenden eine fast ständige Verfügbarkeit erwartet, auch wenn das nicht direkt kommuniziert wird. Außerdem muss gewährleistet werden, dass alle Studierenden Zugang zum Forum erhalten. Das muss seitens des Dozierenden überprüft werden, um eine Gleichbehandlung der Studierenden zu gewährleisten, da über das Forum zusätzlich zur Vorlesung Informationen seitens der Dozierenden verteilt werden.

Diese Art Forum läuft schon seit mehreren Jahren und der rege Gebrauch seitens vieler Studierender zeigt, dass sich dieses Medium bewährt hat.

Eine Möglichkeit, die Briefmethode in größerem Rahmen einzusetzen, wäre die Erstellung eines Wiki zusammen mit den Studierenden. In diesem könnten Studierende die Sie interessierenden Themen für alle technisch dokumentieren.

Zusammenfassung und Ausblick

Planspiel und Briefmethode sind zwei konstruktivistische Methoden, die nach Meinung der Autoren sehr gut für die Ausbildung im Softwareengineering geeignet sind. Die Lernenden lassen sich für das Planspiel einfacher motivieren als für die Briefmethode.
Planspiele im Software Engineering konfrontieren möglichst realistisch mit einer Praxissituation. Die Studierenden können dabei zum kreativen, weitgehend autonomen und selbstorganisierten Handeln in Bezug auf konkrete Probleme und deren Lösung motiviert werden und nehmen dabei unterschiedliche Positionen in einem komplexen Softwareentwicklungsprozess ein.

Die Briefmethode führt zu besserem Verfassen technischer Dokumentation und eignet sich sehr gut, um sich ein Thema anzueignen, oder zu wiederholen. Auch der Spezialfall eines Forums für Studierende und Dozierende wird positiv aufgenommen.

Die Reflexion der Studierenden über den eigenen Lernprozess kann zukünftig durch die Führung eines individuellen Lernjournals unterstützt werden.

Die Dozierenden werden beide Methoden zukünftig häufiger einsetzen. Außerdem sind sie durch die gemachten Erfahrungen mit dem konstruktivistischen Methodenpool hoch motiviert, weitere Versuche mit diesen Methoden für die Software Engineering-Ausbildung durchzuführen.

Literatur

Aviram, A. (2000): Beyond Constructivism: Autonomy-Oriented Education. Studies in Philosophy and Education, 19: 465-489., Kluwer Academic Publishers.

Dewey, J. (1910): How we think (deutsch: Wie wir denken, Zürich 1951).

Hagel, G., Mottok, J., Utesch, M., Landes, D., Studt, R. (2010): Software Engineering Lernen für die berufliche Praxis - Erfahrungen mit dem konstruktivistischen Methodenbaukasten, im Tagungsband des Embedded Software Engineering Kongress' 2010.

Ludewig, J. (2009): Erfahrungen bei der Lehre des Software Engineering, in Jaeger, U. (Hrsg.) und Schneider K. (Hrsg.): Softwareengineering im Unterricht der Hochschulen: SEUH 11, Hannover 2009, dpunkt Verlag.

Macke, G., Hanke, U., Viehmann, P. (2009): Hochschuldidaktik, Lehren, vortragen, prüfen, Beltz Verlag, Weinheim.

Markowitsch, J., Messerer, K., Prokopp, M. (2004): Handbuch praxisorientierter Hochschulbildung, WUV Universitätsverlag, Wien.

Mottok, J., Hagel, G., Utesch, M., Waldherr, F. (2009): Konstruktivistische Didaktik - Ein Rezept für eine bessere Softwareengineering Ausbildung?, im Tagungsband des Embedded Software Engineeing Kongress' 2009, S. 601-610.

Reich, K. (2008): Konstruktivistische Didaktik – Lehr- und Studienbuch mit Methodenpool, 4. Auflage, Beltz Verlag, url: http://methodenpool.uni-koeln.de.

Schmidt, G., Hollweg, G. (2009): Ein integrativer interdisziplinärer Lehrversuch: Softwareengineering und Technisches Schreiben, in Jaeger, U. (Hrsg.) und Schneider K. (Hrsg.): Softwareengineering im Unterricht der Hochschulen: SEUH 11, Hannover 2009, dpunkt Verlag.

Service-Stelle Bologna (2004): Hochschulrektorenkonferenz - Texte und Hilfestellungen zur Umsetzung der Ziele des Bologna-Prozesses an deutschen Hochschulen, Beiträge zur Hochschulpolitik.

Siebert, H. (2009): Selbstgesteuertes Lernen und Lernberatung, ZIEL, Augsburg.

Welbers, U.; Gaus, O. (2009): The Shift from Teaching to Learning, Bertelsmann, Bielefeld.