FH Köln
15.7/16.7.2013

Aktiviert Lernen in der Software Engineering Ausbildung - Erfahrungen mit dem konstruktivistischen Methodenbaukasten

M.Schumm[1], S.Joseph[1], A.Soska[1], M.Reschke[1], I.Schroll-Decker[2], M.Niemetz[1], J.Mottok[1]

Software Engineering Laboratory for Safe and Secure Systems (*www.las3.de*)

[1]Department of Electrical Engineering and Information Technology

[2]Department of Applied Social Sciences

Regensburg University of Applied Sciences

Seybothstr. 2, D–93049 Regensburg, Germany

{michael.schumm, saskia.joseph, alexander.soska, michael.reschke, irmgard.schroll-decker, michael.niemetz, juergen.mottok}@hs-regensburg.de

1. Einleitung

Software Engineering beschäftigt sich mit der systematischen und disziplinierten Entwicklung von Software. Als ausgesprochen volatiles Fach mit einem hohen Anteil an sich ändernden Technologien und Methoden bietet sich diese Fachdisziplin sehr gut an, sowohl im Hochschulkontext als auch in der beruflichen Praxis über geeignete Strategien der Fort- und Weiterbildung und des Lernens selbst nachzudenken.

2. Software Engineering für Mechatroniker

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. 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 bezüglich der über die Programmierung hinausgehenden Schritte bei der Software-Entwicklung. Praktika und Projekte sind für die Lehre von Software Engineering zentral [2]. Die Vorlesung Software Engineering bereitet auf eine schriftliche Prüfung vor. Dagegen wird die Blockveranstaltung Praktikum/Seminar Software Engineering mit einem studienbegleitenden Leistungsnachweis „mit Erfolg“ abgeschlossen.

3. Fachwissenschaftliche Lernziele

Ein wesentlicher Bestandteil des Praktikums/Seminars Software Engineering ist ein Planspiel, welches eine Projektaufgabe mit dem Embedded Roboter System LEGO® Mindstroms® NXT 2.0 beinhaltet. In dieser sollen ein Labyrinth-Algorithmus sowie eine Fernsteuerung- und Fernüberwachung des NXT-Roboters über einen Server-PC und zusätzlich über Web-Clienten, also Browserapplikationen, erstellt werden. Bibliotheken und einfache Beispiele liegen bereits vor. Die für die Projektaufgabe notwendigen Kompetenzen werden vor und während der Durchführung des Planspiels erarbeitet. Im Einzelnen sind diese:

1. Entwicklungsprozess und Methoden
   - Softwareentwicklungsprozess V-Modell 97
     - Projekt-, Qualitäts- und Konfigurationsmanagement
     - System- und Softwareerstellung
   - Requirements Engineering
   - Analyse und Design mit der UML
   - Implementierung
   - Review-Techniken
   - Software-Test
2. Web-Client-Programmierung
   - Webtechnologien (HTML, JavaScript, XML, DOM, CGI, Ajax, Sambar)
3. Server-Programmierung
   - .Net Programmierung mit C++: Basic Windows Forms (Label, Button, Text and Selection Controls)
   - .Net Programmierung mit C++: Graphics using GDI+
4. Embedded System mit NXT-Roboter
   - NXC-Programmierung, Entwicklungsumgebung, Kommunikation
5. Kommunikation zwischen Server und Roboter via XBee

Die Lernzielanalyse wird nach Bloom [5] eingesetzt: Die Formulierung eines Lernziels in der kognitiven Dimension ist „Wissen“ d.h. Lehrinhalte reproduzieren und wiedergeben zu können. Darauf aufbauend folgen zunehmend anspruchsvollere Fragestellungen: Abfragen des Verständnisses, Fähigkeit zur Anwendung, Analyse, Evaluierung sowie auf höchster Ebene die Fähigkeit zur eigenständigen Synthese einer Lösung.

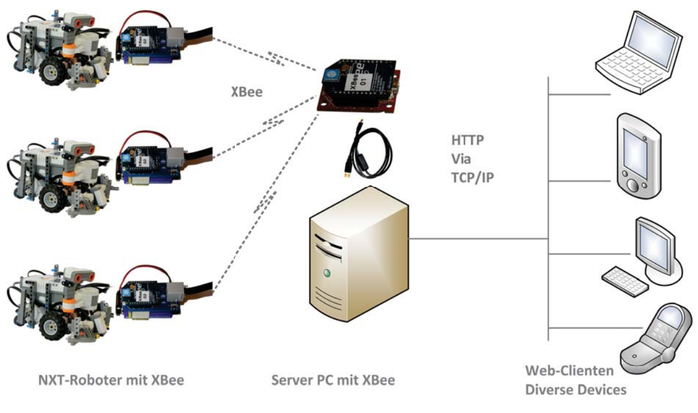

Figure 1: *Grobes Architekturmodell: NXT-Roboter, Server-PC und Web-Clienten*

Aufgabenstellung 1: Labyrinth-Algorithmus
Der NXT-Roboter soll mit Hilfe eines frei wählbaren Algorithmus und Sensor-Unterstützung den Ausgang aus einem Labyrinth (siehe Abbildung 2) finden können. Dabei soll das Labyrinth kartographiert werden, um einen zweiten Roboter, ohne Sensor-Unterstützung, auf dem kürzesten Weg aus dem Labyrinth steuern zu können.

Figure 2: *NXT-Roboter findet selbstständig den Weg aus einem Labyrinth.*

Aufgabenstellung 2: Fernsteuerung und Fernüberwachung
Zusätzlich soll der NXT-Roboter auch von der Applikation auf dem Server und über allen gängigen Browsern auf den Web-Clienten ferngesteuert und fern überwacht werden können.

4. Aufbau der Lehrveranstaltung

Die Blockveranstaltung Praktikum/Seminar Software Engineering findet an fünf Tagen statt. Während in den ersten beiden Tagen Fachthemen erarbeitet und vermittelt werden, werden in den letzten drei Tagen in einem als Projektarbeit ausgelegten Planspiel die Kenntnisse und Fertigkeiten in den Fachthemen vertieft und angewandt.

Bereits zwei Monate vor der Blockveranstaltung findet eine Vorbesprechung mit den Studierenden statt. Hier werden sowohl die Themen für die Seminarvorträge, als auch für die Poster für Open Space vergeben. Den Studierenden werden Literaturhinweise zur Bearbeitung der Aufgaben in der Lernplattform moodle zur Verfügung gestellt.
Die Studierenden werden während des Planspiels in mehrere Projektgruppen mit jeweils ca. 10-20 Teilnehmenden aufgeteilt. Während der Blockveranstaltung Praktikum/Seminar Software Engineering stehen Seminar- und Rechnerräume für die einzelnen Projektgruppen zur Verfügung. An der Blockveranstaltung haben bereits Semestergruppen mit 20 bis 60 Studierenden teilgenommen.

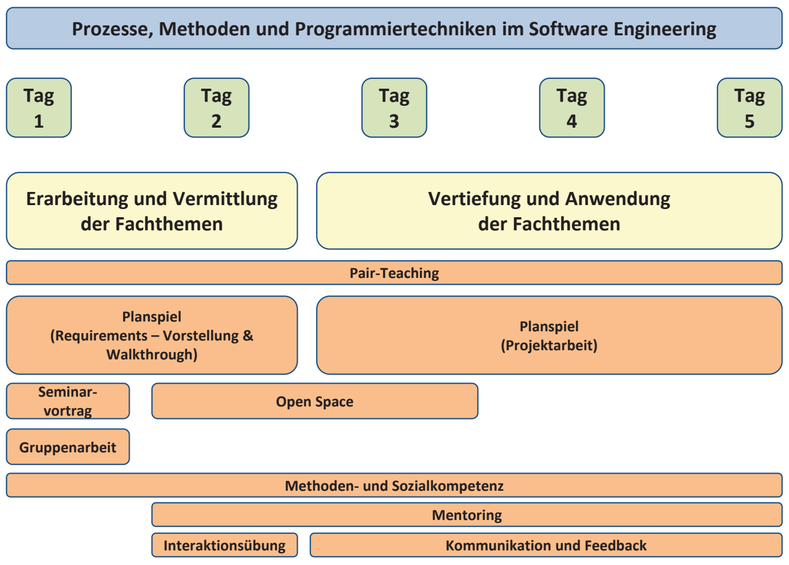

Figure 3: *Aufbau des Praktikums/Seminars Software Engineering*

5. Methodenbaukasten in der Lehrveranstaltung

Der konstruktivistische Ansatz spricht sich dagegen aus, dass Wissen von einem Lehrer an Lernende übertragen werden kann. Der Konstruktivismus setzt auf die Schaffung optimaler Bedingungen für Lernende [8], so dass sie Wissen für sich selbst aneignen können. Anstatt Lernenden Informationen sowie verschiedene Skills zur Verfügung zu stellen, bevorzugt der Konstruktivismus es, ein Umfeld zu fördern, in dem Lernende Wissen durch Erforschen und Untersuchen erwerben, und dies entweder alleine oder in Gruppen [3].
Mit Einsatz von Elementen des Konstruktivistischen Methodenbaukastens [4, 6] und der Aktivierenden Lehre ist der Anteil an Frontalunterricht reduziert. Seit dem WS 2009/2010 sind im Praktikum/Seminar Software Engineering die folgenden Elemente eingesetzt worden:

Elemente	**Lernsituation**
Pair-Teaching	Interdisziplinäres Team vorleben, Verzahnung im Führungstandem, Erlebtes Führungsteam, Konkurrenz versus Zusammenarbeit/Dialog, Klima offener Kommunikation vorleben
Seminarvortrag	Studierende als Lehrer: Durchs Lehren Lernen („Homo docens “)
Gruppenpuzzle	Studierende als Lehrer: Fachthemen zur Durchführung des Planspiels erarbeiten und vermitteln
Open Space	Studierende als Lehrer: Fachthemen zur Durchführung des Planspiels vermitteln
Problemorientiertes Lernen	Ein Brettspiel erarbeiten zum V-Modell 97 [1]
Planspiel	Abwicklung eines Softwareprojekts, Individualität versus Kooperation: Teambildung und Rollenverteilung, Rolle ausgestalten
Mentoring	Studierende höherer Semester geben ihr Wissen weiter
Kommunikation	Verbesserung des Kommunikationsprozesses innerhalb des Projektes lernen
Feedback u.a. Blitzlicht	Rückmeldung zur Prozessverbesserung im Projekt und Reflektion des eigenen Lernprozesses
Methodenkompetenz und Sozialkompetenz	Selbstreflexion durch Gespräche mit den Rolleninhabern während des Projektes (Planspiel)
Interaktionsübung	Z.B. Ballspiel, dabei Aufbau und Verbesserung eines Prozesses, Berücksichtigung unterschiedlicher Wahrnehmungskanäle (Lerntypentheorie), andere Lernformen kennenlernen
Kreativitätstechnik Cyberstorming	Wissen und/oder Informationen über die Programmierung in C bzw. Algorithmen können dabei auf die einzelnen Lernenden aufgeteilt sein, aber die Gruppe erledigt die Aufgabe gemeinsam
Briefmethode	Ein Brief an einen Projektpartner beschreibt eine eigene Problemlösung
Lerntagebuch	Die Studierenden dokumentieren, erkunden, überprüfen und verändern möglicherweise die eigene Lernpraxis
Lernplattform moodle	Zur Erarbeitung von Lerninhalten, als Knowledgebase in der Projektarbeit
Gruppenarbeit	Gemeinschaftliche und selbst organisierte Erstellung von Vorträgen auf Basis von Literaturarbeit

6. Evaluation und Eindrücke WS 2012/13

Im WS 2012/13 nahmen 52 Studenten am Praktikum/Seminar Software Engineering und 49 davon an der Evaluation teil. In den Abbildungen 4 und 5 werden exemplarisch die Selbsteinschätzungen der Studenten bezüglich des Verständnisses der fachlichen Inhalte (im Beispiel Applikation auf Server) nach den Frontalvorträgen durch die Dozenten am Anfang der Projektwoche und am Ende der Projektwoche verglichen. Durch die selbstständige Auseinandersetzung mit den fachlichen Themen ist eine deutliche Verbesserung zu erkennen.

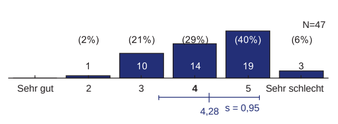

Figure 4: *Selbsteinschätzung Verständnis nach Frontalvortrag*

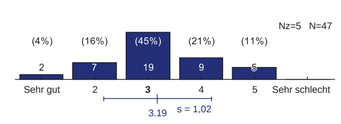

Figure 5: *Selbsteinschätzung Verständnis nach selbstständiger Arbeit*

Die Studierenden bewerten die selbstständige Arbeitsweise (siehe Abbildung 6) sehr positiv und wünschen sich mehr Veranstaltungen wie die hier vorgestellte (siehe Abbildung 7).

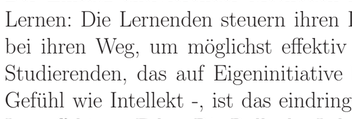

Figure 6: *Antworten auf die Frage: „Wie bewerten Sie das eigenständige Erarbeiten von fachlichen Inhalten?“*

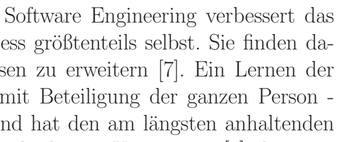

Figure 7: *Antworten auf die Frage: „Es sollen mehr Veranstaltungen wie PSE/PS aufgebaut sein.“*

Eindrücke aus der Veranstaltung in bildlicher Form:

Figure 8: *Open Space: Studierende als Lehrer vermitteln Fachthemen*

Figure 9: *Interaktionsübung: Studierende verbessern ihre Kommunikation untereinander.*

7. Zusammenfassung und Ausblick

Der Einsatz aktivierender Methoden im Fach Software Engineering verbessert das Lernen: Die Lernenden steuern ihren Lernprozess größtenteils selbst. Sie finden dabei ihren Weg, um möglichst effektiv ihr Wissen zu erweitern [7]. Ein Lernen der Studierenden, das auf Eigeninitiative beruht, mit Beteiligung der ganzen Person - Gefühl wie Intellekt -, ist das eindringlichste und hat den am längsten anhaltenden Lerneffekt zur Folge. Die Rolle des Lehrenden wechselt vom Unterweiser [3], der seine Lehre nach den Prinzipien des Vormachens und Nachmachens strukturierte, hin zum Lernberater. Der Lernberater übergibt den Lernenden Aufgabenstellungen und motiviert die Lernenden zum Durchdenken und Durcharbeiten dieser Aufgaben. Dabei begeben sich die Studierenden in die Situation des forschenden Lernens, bestimmt durch Freiheit und Struktur. Die Studierenden erkennen, dass eine Übertragung der erfahrenen konstruktivistischen Methoden auf Lebenslanges Lernen im Berufsumfeld möglich wird.

Literatur

[1] Landes, D., Pfeiffer, V., Sedelmaier, Y., Mottok, J., Hagel, G.: Learning and Teaching Software Process Models, IEEE EDUCON 2012, Marrakesch, Marokko, 2012.

[2] Ludewig, J.: Erfahrungen bei der Lehre des Software Engineering, Software Engineering im Unterricht der Hochschulen, dpunkt.verlag, Heidelberg, 2009.

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

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

[5] Studt, R., Mottok, J., Utesch, M., Landes, D.: Anwendung der Bloom'schen Taxonomie auf Lehrinhalte des Software-Engineerings, Embedded Software Engineering Kongress, Sindelfingen, 2009.

[6] Macke, G., Hanke, U., Viehmann, P.: Hochschuldidaktik, Lehren, vortragen, prüfen, Beltz Verlag, Weinheim, 2009.

[7] Waldherr, F, Walter, C.: didaktisch und praktisch, Ideen und Methoden für die Hochschullehre, Schäffel-Poeschel, Stuttgart, 2009.

[8] Mottok, J., Hagel, G., Utesch, M., Waldherr, F., Konstruktivistische Didaktik- ein Rezept für eine bessere Software Engineering Ausbildung?, Embedded Software Engineering Kongress, Sindelfingen, 2009.

Acknowledgements

Die vorliegende Arbeit ist gefördert durch das vom BMBF finanzierte Verbundvorhaben „Experimentelle Verbesserung des Lernens von Software Engineering (EVELIN)“, Förderkennzeichen 01PL12022F, Projektträger DLR. Das Verbundvorhaben ist im Bund-Länder-Programm für bessere Studienbedingungen und mehr Qualität in der Lehre angesiedelt. Die Hochschulen Aschaffenburg, Coburg, Kempten, Landshut, Neu-Ulm und Regensburg sind Verbundpartner, weitere Informationen unter www.las3.de und www.qualitaetspakt-lehre.de.
Dipl.Päd. Fritz Joas sei gedankt für seine Rolle als Partner in der Methode Pair-Teaching.