veritas
iustitia
libertas
Freie Universität Berlin

Seminar " Ausgewählte Beiträge
zum Software Engineering"

Software Engineering Projects

Christopher Özbek

Freie Universität Berlin, Institut für Informatik
http://www.inf.fu-berlin.de/inst/ag-se/

Überblick

- Forschungsergebnisse der letzten vier Jahre
  - "Conference on Software Engineering Education and Training (CSEET)" hauptsächlich
  - "ACM Special Interest Group on Computer Science Education (SIGCSE)" ein bisschen
  - Eine Ausgabe von GI Softwaretechnik-Trends (für die deutschen Beiträge)
  - FU Berlin
- Vorstellung der vielfältigen Ansätze zur Durchführung eines Projektkurses im Bereich Software-Engineering.
- Dies ist auch der Schwerpunkt der CSEET-Beiträge (50%). Bei der Lehre im Bereich SE gibt es hauptsächlich Beiträge zur Curriculum-Diskussion.

Wieder etwas zurück...

- Problematik der Ergebnisse:
  - Wenig empirisch fundiert
    - Kleine Datenbasis
    - Keine richtigen Studien
  - Keine Berücksichtigung von externen Einwirkungen (Hawthorne).
  - Extremer positiver Bias gegenüber der eigenen Lehrmethode.
  - Die Studenten fanden auch immer alles toll was mit ihnen angestellt wird.
- Also eher alles als Ideensammlung für Software- und Softwaretechnikpraktika zu verstehen. Daher stammt ja auch der initiale Impuls.

Wieso eigentlich Projekte/Praktika?

- *"Theoretisches, aber nicht eigenständig erfahrenes Wissen führt nicht immer zu entsprechenden Handlungen. Oft kommen die erst, wenn 'etwas passiert ist' oder das theoretische Wissen praktisch durchgespielt wurde."* [AWS Übungsblatt Tschernobyl]
- Erlernen von
  - Soft-Skills
  - Prozessen
  - Technologien

  mit Relevanz für den beruflichen Alltag als Softwaretechniker v.a. in der Industrie.
- **Transfer** üben (in der Lage sein, Erlerntes auf neue Situation/Problemstellungen anwenden zu können)
- Für die Probleme sensibilisiert werden *("Gegen die Wand fahren"*-Ansatz).

Projektformen (I)

Aus [FordG91] "SEI Report on Graduate Software Engineering Education":

- Capstone project (Gesellenstück)
  - Studenten schließen die Ausbildung mit einem Softwareentwicklungsprojekt ab. (Seattle University, Monmouth College, Texas Christian University)
- Fortlaufendes Projekt
  - Studenten nehmen an einem Projekt teil, welches von Jahr zu Jahr fortlaufend betrieben wird (die Software Fabrik) und entwickeln und verbessern Software Engineering Werkzeuge und Umgebungen. (University of Southern California)
- Koordiniertes Projekt über mehrere Kurse hinweg
  - Ein Projekt wird in vier Kursen (Softwareanalyse, Design, Test und Wartung) verwendet, wobei die Studenten die Kurse in beliebiger Reihenfolge hören dürfen. (Arizona State University)

Projektformen (II)

- Industriekooperation
  - Ein Jahr Lehre wird durch sechs Monate in der Industrie in einem professionell gemanagten Software Projekt ergänzt und der Abschluss durch ein Projektsemester oder eine Masterarbeit basierend auf den Arbeitserfahrungen erreicht. (University of Stirling)
- Kommerzielle Softwarefirma
  - Studenten nehmen an Projekten einer Kommerziellen Softwarefirma teil, welche als Kooperation zwischen den Colleges und lokalen Firmen gegründet wurde. (Imperial College)
- Design Studio
  - Studenten arbeiten an einem Projekt unter dem direkten Auge eines erfahrenen Software Designers, in einer Meister/Lehrling-ähnlichen Beziehung. (Carnegie Mellon University)
- Software Hütte (ein kleines Software Haus)
  - Studenten erarbeiten Module und "verkaufen" diese schließlich an andere Studenten, um Projekte zu erstellen, welche an den Übungsleiter "verkauft" werden. (University of Toronto)

Entwickeln, Warten, Reverse Engineering

- Entwickeln:
  - Folgt dem "echten" Verlauf von neuen Projekten.
  - Nachteil: Der kurze Zeitrahmen des akademischen Semesters verhindert die Entwicklung von großen Projekten.
- Weiterentwicklen, Warten:
  - Existierende Codebasis, grundsätzliche Entwurfsentscheidungen schon getroffen.
  - Lernen aus den Strategien der Vorentwickler.
  - Test-getriebene Entwicklung bietet sich an.
  - Bsp: [AllCarRei03] und Dr. Java Entwicklungsumgebung, die inkrementell von Jahrgängen weiterentwickelt wird.
- Reverse Engineering:
  - [BotheK01] versucht dies von Wartung abzutrennen, indem er höheren Wert auf Analyse und abstrakte Darstellung des Systems legt.

Glaubwürdigkeit

- Viele Autoren betonen die Wichtigkeit, den Studenten klar zu machen, dass ihre Arbeit nützlich ist und in der "wirklichen" Welt gebraucht wird.
- Der größte Feind diesen Anliegens sind akademische Spielzeugaufgaben (toy projects), welche nach abgeschlossenen Semester unbeachtet auf einer Webseite versauern.
- Natürlich gibt es unzählige andere Probleme, welche durch die Einbindung von externen Parteien in das Projekt entstehen.
- Problem bei der Erstellung von "Real-World"-Projekten: Problemstellungen können nicht ohne Verlust von Glaubwürdigkeit wieder verwendet werden.

Industrie vs. Open Source [AndLut00]

- Beide Optionen stellen einen Bezug zu "echten" Kunden her.
- Industrie:
  - Projekte kommerziell wertvoll für den Kunden
  - Cutting-Edge-Technologie
  - Direkter Kunden-Kontakt
  - Höheres Risiko des Abbruchs (=> Halb formelle Absprachen können hier helfen)
- Open Source:
  - Command Line Utilities sind nicht sonderlich interessant im Vergleich zu "klick-i-bunt-i".
  - Informationen auf den Webseiten müssen nicht immer aktuell sein, d.h. es gilt vorsichtig zu sein, wenn man ausgeschriebenen Aufgaben von dort übernimmt (nachfragen).

Projekt-Ideen

- Online Kursverzeichnis, Mikroprozessorsimulator Terminverwaltung, PDA Anwendung für Bestandsaufnahmen. [SebernMJ02]
- Text-Editoren und Text-Formatierer nicht, sondern Spiel. [HenryS83]
- Webshop, Spiele, CMS. [FU Berlin]
- Volltextindizierung auf PCs. [Uni Karlsruhe]
- Riesige Projekt-Bibliothek (>300)
  - http://www.cs.colorado.edu/ugrad/seniorproject/
- http://www.comp.utas.edu.au/units/kxa351/ z.B.:
  - Heritage Assessment tool, Book Seller Webshop, Interactive Dog, Surgical Instrument Training Project, VoIP

Einsatz von Rahmenwerken

- Rahmenwerke können in frühen Projekten, in denen noch viel über die Programmierung gelernt werden muss, hilfreich sein. [DemFisHus02]
  - Beispiel: SalesPoint der TU Dresden
- "Most programming labs assign students to build a dog house - they get to bang together the entire product but they don't need to engineer it. We think a better approach is to situate students within the skeleton framework of a real house and let them build small pieces of an preengineered product." [DalbeyJ98]
- Die gegenteilige Argumentation sagt natürlich, dass man in einem vorgegebenen Haus nur noch Steine aufeinander schichtet und vom architektonischen Entwurf nichts mitbekommt.

Gruppenbildung

- Von den Studenten selbst bestimmt:
  - Weniger Durchmischung und Kennenlerneffekte.
  - Dauert länger und am Schluss müssen die verbleibenden Studenten zugeteilt werden.
  - Eine existierende Kerngruppe, welche zusätzliche Mitglieder aufnehmen muss, kommt in eine Vorteilssituation gegenüber den "aufgenommenen" Studenten.
- Vorgegeben:
  - Rechnergestützte Gruppenbildung: Ähnliche Fähigkeiten (bereits besuchte Kurse), ähnliche Arbeitspensen (Kurse welche gerade besucht werden), Angabe von Freunden und Verfügbarkeit von mindestens zwei Terminen für Gruppentreffen. [HenryS83]
- Zufällig
  - Entspricht eher den Anforderungen in realen Arbeitsumfeldern.

Gruppengröße

- Ganz unterschiedlich in allen Papers
- Schwerpunkt liegt bei 4 - 10
- Rund 10% sind dann im Bereich 10 - 20 angesiedelt.
- Keines der Paper begründet diese Wahl oder erwähnt nennenswerte Auswirkungen dieser Entwurfsentscheidung.
- Ausreißer:
  - 60 (sic!) [BroBrü99] + ein Kunde

Kunden

- 1.) Aus der Industrie
  - + Realistischere Anforderungen an die Studenten
  - - (Unrealistische) Erwartungen der Kunden werden leicht enttäuscht
  - - Abhängigkeit von der Einsatzbereitschaft der Kunden (d.h. Fragen der Verlässlichkeit müssen angesprochen werden, bevor der Kunde mitten im Semester ein Projekt abbricht [AndLut00])
  - => Semi-Förmliche Übereinkünfte machen Sinn
  - - Probleme des geistigen Eigentums und Geheimhaltung
  - - Ausrichtung am akademischen Kalender / Zeitplanung
  - => Abwandlung: Wissenschaftlichen Forschung
- 2.) Der Dozent selbst
  - - Zeitaufwand
- 3.) Studenten in höheren Kursen
- 4.) Tutoren

Rollen

- Ein Teil der Projekte orientieren sich an der Vergabe von (teils rotierenden) Rollen:
  - Team Lead, Architect, Testing, Configuration
  - Team Leader, Development Manager, Planning Manager, Support Manager, and Quality/Process Manager. [SebernMJ02]
- Dies soll dem Arbeitsalltag angelehnt sein.
- Fortlaufende Entwicklungsprojekte seien aber eher für problemorientierte Unterteilungen geeignet, da so passend portionierte Teilaspekte ausgewählt werden können. [BotheK99]

Wahl der Programmiersprache

- In den meisten Projekten von außen vorgegeben.
- Aufgetaucht in den Papers: Java, Eiffel, Smalltalk, C++.
  - Keine Skriptsprachen.
  - Java klarer Gewinner im universitären Bereich.
- Ist aber selten überhaupt ein Punkt der Diskussion.
- Diese Frage scheint eher für Programmierkurse wichtig zu sein und bei der Lehre von Datenstrukturen (z.B. Transition von Modula nach Java [CowlingAJ01]).
- Viel Lamento über Java ist mittlerweile mit Version 1.5 hinfällig geworden (v.a. durch Generics).

Dauer

- Ausgangsprobleme mit einsemestrigen Projekten:
  - Studenten beginnen bei Null und machen wohl Jahr für Jahr immer die gleichen Fehler.
  - Lange Anlaufphasen in Rahmen von Industriekooperationen.
  - Was nicht zum Semesterende fertig wird, wird nie fertig.
  - Keine Wartung und Weiterentwicklung für Industriepartner möglich.
  - Daher werden oft nur Spielzeug-Projekte ausgewählt, die anschließend in der Schublade verschwinden.
- Mit längeren Projekten lassen sich Teile dieser Probleme lösen.
- Lösungsvorschlag von SebernMJ02:

Software Development Labs [SebernMJ02]:

- *"large-scale, ongoing projects in the context of a standardized and evolving development process."*
  - Durch überlappende Neubesetzung lässt sich Wissensweitergabe unter Studenten stärken und auch größere Projekte über längere Zeiträume realisieren. => "Open Houses", "Training", "SE Management Class"
  - Umgekehrte Reihenfolge der getätigten Prozessschritte. Am Anfang ist man Programmierer in existierenden Projekten und wirkt später bei der Spezifizierung und dem Entwurf von neuen Projekten mit. => "Grave to Cradle"
  - Neben der Entwicklungstätigkeit besteht ein Prozess zur fortlaufenden Prozessverbesserung und –anpassung. => "Process Improvement Proposal" (PIP)
  - 65% Entwicklung, 30% Organisation, 5% Weiterbildung
  - Ähnlich: Georgia Institute of Technology Real World Lab

Learning by Failing [BerKlaKel02]

- Ausgangspunkt: Lernen durch Fallstudien
  "The problem with this approach, in our experience, is that reciting or discussing case histories in class causes students only to laugh at the stupidity of others"
- Idee: Studenten gezielt scheitern lassen
  - Durchlebte Fallstudien (Live-Thru Case Histories)
- Anschließend weitere Fallstudien besprechen
- Und endgültig Verständnis für Fallstudien abprüfen:
  - "In a Live-Thru Case History Exam students are given a [...] case history which, at a critical point, resulted in a mysterious failure. Students are then given a series of questions, one at a time, having to do with steps in identifying and correcting the reasons for the failure."

Motivation and employability [RatWooTho02]

- University of Wales, Aberystwyth
- Vor dem ersten Semester wird ein Gruppenerlebnis (shared group experiences) durch eine Projekt- und Abenteuerwoche hervorgerufen. Dies wird eng von den Professoren betreut, die hiermit den Kontakt zu den Studenten initiieren.
  - Bsp: Das "Trapez" = Sprung von einem 10 m hohen Mast zum Ergreifen eines freihängenden Trapezes.
  - "If I can handle the trapeze, then I can handle a course on Enterprise Java Beans!"
- Bereits im zweiten Semester werden Studenten angehalten, mit eingeladenen freiwilligen Vertretern aus der Industrie Bewerbungsgespräche und die Gestaltung von Lebensläufen zu üben.

Programmierbörse [HenryS83]

- Zur Trennung von Design und Implementierung werden spezifizierte Module auf der Programmierbörse für virtuelle Geldeinheiten angeboten.
- Die Studenten können dann Programmieraufgaben annehmen und hierdurch Punkte für ihre Kursnote erwerben.
- Die Projektteams bewerten die Komponenten und erteilen Abzüge für Abweichungen von der Spezifikation, fehlende Tests oder Verspätungen oder vergeben extra Punkte für Module, welche sich als komplexer als angenommen herausstellten.
- Ähnlich der "Software Hütte" [FordG91]

Paarprogrammierung in Programmierklassen

- Hauptprobleme:
  - Terminfindung (das amerikanische System mag hier eine größere Rolle spielen)
  - Einstellung der Teilnehmer
- Lösungsvorschläge:
  - Verpflichtende Paarprogrammierung unter Aufsicht
  - Paare klar nach passenden Fertigkeitsstufen auswählen
  - Kodierrichtlinien bezüglich Stil vorgeben
- [BevWerMcD02]
- Zu XP an der Uni: [SchJoh03]

Evaluierung der Arbeit (I)

- Problem: Wie wird die individuelle Leistung gegenüber der des Teams abgegrenzt?
- Studenten bewerten sich selbst
  - Die Studenten bewerten die Zeitaufteilung innerhalb des Projektes vertraulich, und aus dem durchschnittlichen Wert errechnet der Übungsleiter einen Wert für den jeweiligen personenbezogenen Betrag. [HenryS83]
  - Verschiedene andere Techniken [WilLaw01]:
    - Skalen-Bewertungen (sehr gut – schlecht, sog. "Likert"-Skala)
    - Ranking (verteile €1000)
    - Paarung (Entscheider, Macher, Richter, Zuschauer…)
    - Satzvervollständigung (Die Stärke des Teams war____)
    - Entwickler-Tagebuch
  - Diese Bewertungen glätten allerdings die wahren Leistungen, so dass schwache bevorteilt und starke Studenten benachteiligt werden. [GooHor02]

Evaluierung der Arbeit (II)

- Problem: Wie kann die Qualität der Programme ermittelt werden? [MenUla99]
- Dynamischer Test / Datenabdeckung
  - Programm wird ausgeführt und stichprobenhaft oder mittels Testsuite gemäß der Anforderungen geprüft.
  - Problem: Auch schlecht entworfene und implementierte Projekte schlüpfen durch die Maschen.
- Statische Analyse
  - Der Übungsleiter sieht den Programmtext und eingereichte Dokumente durch und benotet die Qualität entsprechend.
  - Der Programmtext kann auch automatisch ausgewertet werden mittels Metriken (Testabdeckung, Größe von Funktionen, Anzahl der Kontrollpfade, Halstead = Anzahl von Operatoren, McCabe Cyclomatic Complexity Value = Anzahl der logischen Prädikate, Bezeichnerlänge, etc.).

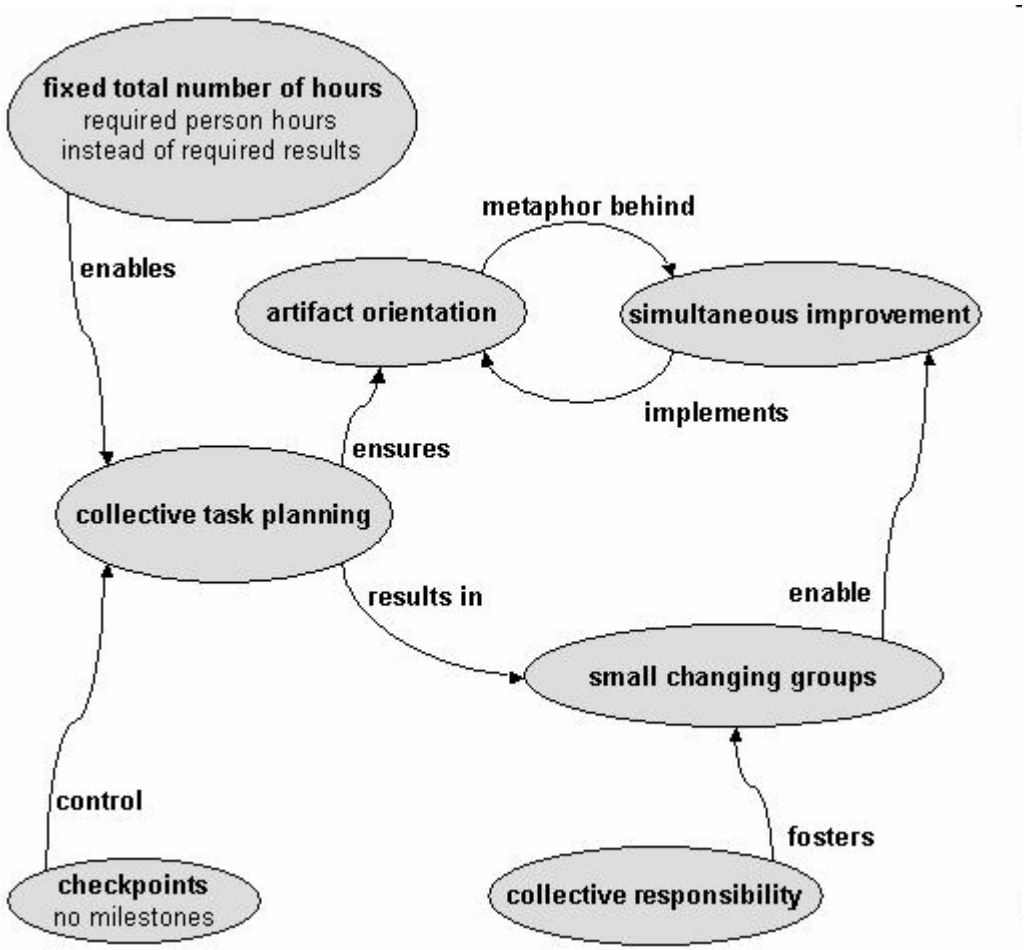
Die FU Perspektive [LoehrKP99]

- SE sei zu schwammig und wird deshalb nicht ernst genommen.
- Prozesse würden von den Studenten nur als bürokratische Belastung empfunden.
- Werkzeuge/Technologien werden angenommen aber Prinzipien/Verfahren nicht.
- Lösungvorschlag:
  - Zwei-semestriges verpflichtendes SWT-Projekt.

Software Praktikum an der FU [DraheimD04]

- Namensgebung eher unglücklich, besser Programmier- und Technologiepraktikum
- Bisherige Gruppengröße: 15-20
- Ab 2004 geplant: 7
- Ein vorgegebener vertikaler Prototyp (Spike) soll ausgebaut werden
- Berührungspunkte mit praxisrelevanten Technologien:
  - SQL
  - J2EE / Application-Server
- Verwendeter Prozess: "Education for Actual Software Engineering (EASE)"

EASE



Quelle: http://www.inf.fu-berlin.de/projects/ease/process/

Andere deutsche Hochschulen

- Dortmund [GruhnV99]
  - 6 Wochen Pflichtpraktikum im Grundstudium - Kleine Tätigkeiten der Softwareerstellung
  - Zwei-semestriges Projekt (8 SWS) im Hauptstudium als Projektgruppen (8–12) – Vordefinierte Aufgabe
  - Vorlesung ist "rollenbasiert" aufgebaut
- Karlsruhe [meine Erfahrungen]
  - Weder Programmierkurse noch verpflichtende Softwarepraktika
  - Programmierkenntnisse "materialisieren" im Grundstudium
  - Wahlpflichtfach SWT
- Hamburg [ZüllighovenH99]
  - Ähnliche Konzeption wie Karlsruhe
  - Ein(e) Workshop/Vorlesung zum Thema OO Softwareentwicklung existiert.

Kleinere Bonmots

- Den Kurs zweimal direkt hintereinander anzubieten, so dass Studenten die Wahl des Termins haben, führt dazu, dass die Abneigung gegen das Programmieren im zweiten Kurs höher ist. [DicPosMil01]
- Gruppenprojekte tendieren dazu, dass unterschiedliche Gruppenmitglieder auch Unterschiedliches lernen. [DanFauNew02]
- "real-world loss of a team member" – Der produktivste und unentbehrlichste Entwickler wird aus dem Projekt in der Mitte der Zeit entfernt und in ein anderes Team verfrachtet. [BerKlaKel02]
  - Lässt sich noch Steigern wie bei [BroBrü99]: komplette Übergabe des bisherigen Fortschritts an ein anderes Team inklusive aller Artekfakte.
  - Eine Sammlung solcher "dirty tricks": [DawsonR00]

Kleinere Bonmots (II)

- "In one of the GNU projects, there was more interest expressed in the document than in the code produced, since the document described the organization of the code." [AndLut00]
- "[...] this author has seen no case where a peer evaluation was used to criticize a students technical ability – only their lack of effort." [VaughnRB00]
- Wählen die Studenten durch Mehrheitsentscheidung ein Projekt, so kann dies immer dazu führen, dass die Minderheit nicht wirklich motiviert wird. [AndLut00]
- "To treat the domain experts with care, we need a staff member acting as an intermediate between the students and the domain experts. Otherwise, domain experts may easily be overwhelmed by excessive inquiries (...)" [BotheK01]

Sonstiges

- Viel wichtiger sind für Softwaretechniker sowieso die Grundlagen (Automaten, Komplexitätstheorie, …). [ParSol02]
- Anstatt großer Präsentationen wurden eine Vielzahl von Kleinstseminaren mit je einem Mitglied jeder Gruppe abgehalten. Damit wird die Partitionierung durch die Gruppenaufteilung wieder etwas aufgeweicht und Probleme mit der Leistung einzelner fallen leichter auf. [AndLut00]
- Wenn man will, dass Studenten in Projekten weiche Fertigkeiten (z.B. Teamarbeit, Konfliktlösung, effektive Kommunikation) lernen, dann sollte man dies ihnen auch direkt beibringen. [GooHor02]

Pointer-Artikel in Themengebiete

- "Gesamtsicht" => [SebernMJ02]
- "Wartung in Softwaretechnik-Praktika" => [BotheK01]
- "Pädagogischer/Kognitiv" bzw. klar konstruktivistisch => [RatThoWoo01]
- "Open Ended Group Projects" => [DanFauNew02]
- Es gibt aber noch viel mehr...
- Wer mehr wissen will über SE-Lehre:
  - [DugTho02] - An Historical Investigation of Graduate Software Engineering Curriculum.
  - [HalZusKöh02] - Teaching the Unified Process to Undergraduate Students.
- Alle Artikel unter \\rhein\agse\home\oezbek\papers

Danke!