Softwaretechnologie in der Informatikausbildung
an der
Universität Dortmund
-
Ein Erfahrungsbericht

Volker Gruhn
Informatik
Lehrstuhl für Software-Technologie
Universität Dortmund
gruhn@ls10.informatik.uni-dortmund.de

1. Ist-Situation

Die softwaretechnologische Ausbildung begleitet das gesamte Informatikstudium an der Universität Dortmund. Sie umfasst Lehrveranstaltungen wie Vorlesungen und Übungen und praktische Anteile in Form eines Praktikums und einer studentischen Gruppenarbeit. Im Rahmen des Grundstudiums ist vor allem das Softwarepraktikum (SoPra) zu nennen. Dieses interne Praktikum dauert sechs Wochen, findet in der Regel in den Semsterferien statt und ist eine Pflichtveranstaltung (das heisst, alle Studierenden müssen dieses Praktikum absolvieren). Im SoPra arbeiten kleine Gruppen von Studierenden zusammen, um Grundmerkmale der Objektorientierung an einigen kleinen Aufgaben zu erlernen. Hierbei wird ein wasserfall-orientierter Softwareprozess befolgt. Die zu erledigenden Aktivitäten beschränken sich auf die Softwareerstellung. Begleitende Aufgaben aus den Bereichen des Projektmanagements, des Qualitätsmanagements und des Konfigurationsmanagements werden ausgeklammert, so dass die Studierenden im wesentlichen das Programmieren-im-Kleinen üben.

Im Rahmen des Hauptstudiums gibt es die zur Gruppe der praktischen Vorlesungen gehörende Softwaretechnologie-Vorlesung. Sie hat einen Umfang von sechs Semesterwochenstunden (vier Vorlesungsstunden, zwei Übungsstunden) und ist eine Wahlveranstaltung. Ziel der Veranstaltung ist es, den Studierenden einen Überblick über Prinzipien, Methoden, Sprachen und Werkzeuge der Softwaretechnologie zu vermitteln.

Als Pflichtveranstaltung im Hauptstudium gibt es die sogenannten Projektgruppen. In einer Projektgruppe arbeiten acht bis zwölf Studierende unter der Anleitung von zwei bis drei Betreuern für die Dauer von zwei Semestern zusammen, um eine von den Betreuern definierte Aufgabe zu bearbeiten. Der geplante Umfang einer Projektgruppe beträgt acht Semesterwochenstunden. Diese Planung wird jedoch häufig und insbesondere gegen Projektgruppenende überschritten. In den allermeisten Fällen handelt es sich bei der zu bearbeitenden Aufgabe um eine praktische Aufgabe, so dass ein erheblicher Teil der Projektgruppenzeit mit Softwareentwicklung verbracht wird. Die einzusetzenden Verfahren und Methoden werden von den einzelnen Projektgruppen selbst festgelegt.

2. Erfahrungen mit der Vorlesung Softwaretechnologie

Von der Zielsetzung ausgehend den Studierenden in der Stammvorlesung einen breiten Überblick zu vermitteln und ihnen zudem zu verdeutlichen, dass Softwareentwicklung nicht im wesentlichen aus Programmierung besteht, habe ich die Vorlesung in den Wintersemestern 97/98 und 98/99 mit dem Untertitel „Rollenbasierte Softwareentwicklung“ gehalten.

Grundidee hierbei ist es, nicht eine bestimmte Methode in den Mittelpunkt der Vorlesung zu rücken, sondern statt dessen die an der Softwareentwicklung beteiligten Rollen zu betrachten und die für diese Rollen relevanten Prinzipien, Methoden, Sprachen und Werkzeuge im Kontext der einzelnen Rollen zu diskutieren. Zielsetzung ist es, die Studierenden mit den nötigen Kenntnissen auszustatten, so dass sie nach Abschluss ihres Studiums dauerhaft zur Lösung des Grundproblems der industriellen Softwareentwicklung, nämlich der kalkulierbaren und wirtschaftlichen Entwicklung von Software, beitragen können. Als Rollen werden hierbei unter anderen die des Anforderungsanalytikers, des Spezifizierers, des Designers, des Programmierers, des Testers, des Konfigurationsmanagers, des Qualitätsmanagers, des Projektmanagers, des Vertrieblers und die des Software-Prozessverbesserers diskutiert.

Zu jeder Rolle wird die Grundaufgabe dieser Rolle erörtert, es werden die Wechselwirkungen mit anderen Rollen diskutiert und es werden die benötigten Hilfsmittel vorgestellt. So wird beispielsweise für den Projektmanager erörtert, wie er einen Projektplan zu erstellen und zu verfolgen hat, aber auch einige Grundsätze der Mitarbeiterführung vorgestellt. Für den Spezifizierer werden eine ganze Reihe von Spezifikationsansätzen vorgestellt. Diese reichen von algebraischer Spezifikation über Datenflussdiagramme zu Petrinetzen und UML als Beispiel einer objektorientierten Spezifikationssprache. Bei der Vorstellung formal definierter Spezifikationsansätze (algebraischen Spezifikation, Petrinetzen) werden deren formale Grundlagen erörtert. Die Vorstellung von UML wird durch eine allgemeine Einführung in die Prinzipien der Objektorientierung abgerundet. Es wird problematisiert, dass es für die UML keine formal definierte Semantik gibt. Es werden eine Reihe von UML-Werkzeugen präsentiert. Die unterstützenden Übungen werden nicht werkzeuggestützt durchgeführt. Dies hat einerseits den Nachteil, dass die Studierenden nicht die Gelegenheit haben, sich mit einzelnen Werkzeugen vertraut zu machen, gewährleistet andererseits aber auch, dass die Frage der Werkzeugbeherrschung nicht die nach der sinnhaften Anwendung einer Sprache verdrängt.

Neben der Vorstellung eines Korbes von Methoden und Verfahren wird immer wieder diskutiert, unter welchen Rahmenbedingungen, welche Methode zum Einsatz kommen sollte und wie eine einzelne Methode mit anderen zusammenpasst. Die Struktur der Vorlesung ergibt sich aus der Orientierung an Rollen. Nach einigen einleitenden Bemerkungen ist der wesentliche Teil der Vorlesung in Abschnitte aufgeteilt, die jeweils eine Rolle erörtern.

Diese Strukturierung der Vorlesung bietet aus meiner Sicht einige Vorteile:

- Die Studierenden erkennen, dass die richtige Sprache und das richtige Werkzeug nicht ausreichen, um die Chance auf eine erfolgreiche Softwareentwicklung maßgeblich zu erhöhen. Sie erkennen darüber hinaus, dass es die richtige Sprache und das richtige Werkzeug gar nicht gibt (und auch nicht geben kann), sondern dass unterschiedliche Rahmenbedingungen unterschiedliche Vorgehen und Hilfsmittel erfordern.

- Ein weiterer Vorteil ist, dass die Diskussion der Wechselwirkungen zwischen den einzelnen Rollen verdeutlicht, dass es in der Softwaretechnologie um bewährte Methoden und Verfahren geht und dass diese formal sein können aber nicht müssen. Dies macht explizit, dass die Softwareentwicklung von einem möglichst ingenieurmäßigen Vorgehen abhängt und nur in kleinen Teilen eine mathematisch präzise Wissenschaft ist.
- Der Mix von konkreten Sprachen und Ansätzen einerseits und der Diskussion einiger formaler Grundlagen andererseits gewährleistet, dass die Studierenden Wissen vermittelt bekommen, das direkt anwendbar und für den einzelnen unmittelbar nützlich ist und dass die Studierenden das nötige Rüstzeug erhalten, um zukünftige Methoden und Sprachen verstehen, einordnen und bewerten zu können.

Diesen Vorteilen stehen jedoch auch eine Reihe von Schwierigkeiten gegenüber:

- Das Vorstellen eines Korbes von Möglichkeiten überfordert einige Studierende. Für sie wäre es zweifelsfrei einfacher zu lernen, dass UML (oder irgendwas anderes) die Lösung aller Softwaretechnologie-Probleme bedeutet.
- Die Diskussion einiger weicher Faktoren (gerade im Kontext der Wechselwirkungen zwischen Rollen) nimmt sehr leicht einen anekdotenhaften Charakter an, der dann stark mit der Erörterung formaler Sachverhalte kontrastiert. Dies mag für den ein oder anderen Zuhörer eine willkommene Abwechslung sein. Erfahrungsgemäß stellt es viele Studierende jedoch vor Schwierigkeiten, die nicht auftreten würden, wenn ein solcher Wechsel zwischen harten und weichen Vorlesungsanteilen vermieden werden könnte.

3. Erfahrungen mit Projektgruppen

In meiner letzten Projektgruppe wurde der Ansatz der rollenbasierten Softwareentwicklung angewendet. Das heisst, es wurden zunächst einige Rollen definiert (im konkreten Fall die Rollen Projektmanager, Qualitätsmanager, Systemtechniker, Konfigurationsmanager, Entwickler (als Synthese von Designer und Programmierer)) und diese für bestimmte Zeiträume den Teilnehmern zugeordnet (ein Teilnehmer konnte mehrere Rollen übernehmen, eine Rolle konnte mehreren Personen zugeordnet werden). Die Rollen wurden vierteljährlich rotiert (wobei dies nicht immer ganz konsequent durchgehalten wurde).

Fünf der zwölf Teilnehmer hatten zuvor die Softwaretechnologie-Vorlesung verfolgt. Während es allen möglich war, die benötigten Hilfsmittel (im konkreten Fall UML und Java) relativ schnell zu erlernen, stellte der Ansatz der rollenbasierten Softwareentwicklung höhere Anforderungen. Zwar war es noch möglich, die Rollen zu definieren, aber in der verantwortlichen Wahrnehmung der übernommenen Rollen gab es einige Unsicherheiten. Bei allen Teilnehmern fiel durchgängig die sehr starke Fixierung auf technologische Fragestellungen und die zu geringe Konzentration auf das zu lösende Problem auf. Dies konnte durch die Einführung der Rolle „Marketier und Kundenbetreuer“ deutlich verbessert werden, denn hierdurch geriet die Frage, zu wessen Nutzen denn überhaupt welche Software entwickelt werden sollte stärker in den Mittelpunkt der Betrachtung.

Bei den Teilnehmern ohne Kenntnis der Softwaretechnologie-Vorlesung war das Verständnis für Softwareentwicklung als komplexer und kooperativer Prozess meist nur gering ausgeprägt. Hier musste mühevolle Überzeugungsarbeit geleistet werden, um beispielsweise durchzusetzen, dass vor der Programmierung zunächst ein Entwurf erstellt wird, dass dieser von der gesamten Projektgruppe abgenommen wird, dass Testfälle schon im Entwurf abgeleitet werden, dass Konfigurationseinheiten explizit festgelegt werden und dass der Qualitätsmanager entscheidet, ob die Qualität einer Software reicht, um ein Release zu erstellen.

4. Erfahrungen mit Diplomarbeiten

Bei der Durchführung von praktischen Diplomarbeiten fällt immer wieder auf, dass sich auch solche Studierende für eine praktische Arbeit im Kontext der Softwaretechnologie interessieren, die in ihrem bisherigen Studienverlauf nicht mit softwaretechnologischen Fragestellungen in Berührung gekommen sind (ausser im SoPra). Einige andere haben in der ein oder anderen Weise eine bestimmte Spezifikations-, Entwurfs- oder Programmiersprache kennengelernt und sind der Ansicht, dass genau diese Sprache für jedes Problem benutzbar ist. Erhebliche Mängel bestehen fast durchgängig bezüglich der Frage, wie das Ergebnis einer Softwareentwicklung zu dokumentieren ist.

Interessanterweise stehen auch viele Studierende, die sich während ihres Studiums mit softwaretechnologischen Fragestellungen beschäftigt haben, bei der Anfertigung einer praktischen Diplomarbeit vor Schwierigkeiten. Oft scheint dies darauf zurückzuführen zu sein, dass es erstmals zu einer selbständigen Auseinandersetzung mit einer Softwareentwicklung in einer relevanten Größenordnung kommt. Erwartungsgemäß kommt es fast immer zu erheblichen Unterschätzungen des zu leistenden Entwurfs-, Realisierungs- und Testaufwands.

5. Bewertung

Als eine wichtige Erfahrung der Lehrveranstaltungen in den letzten beiden Jahren ist festzustellen, dass Studierende ihr Studium erfolgreich abschliessen können, ohne jenseits des Programmierens-im-Kleinen mit den Grundprinzipien der Softwareentwicklung in Berührung gekommen zu sein.

Auf der anderen Seite bieten die zahlreichen Wahl- und Gestaltungsmöglichkeiten genauso die Gelegenheit durch die Wahl von aufeinander abgestimmten Standardvorlesungen, Vertiefungsvorlesungen, Seminaren, Projektgruppen und Diplomarbeiten ein einzelnes Gebiet der Informatik zu vertiefen. Einer der wählbaren Schwerpunkte ist die Softwaretechnologie.

Softwaretechnologie kann durch Spielzeugübungen nicht wirklich erlernt werden. Erlernen lassen sich Syntax, Semantik und Pragmatik von Spezifikations-, Entwurfs- und sonstigen Sprachen. Erlernen lässt sich auch die Handhabung eines Werkzeuges. Wenn dieser Lernerfolg nicht kontinuierlich in den Kontext eines gesamten Entwicklungsprozesses eingeordnet wird, entsteht jedoch oft der Eindruck, dass sich Softwaretechnologie in der Beherrschung von Sprachen und Werkzeugen erschöpfen würde. Viele Absolventen verlassen die Universität mit genau diesem Eindruck. Der dann eintretende Praxisschock kann universitätsseitig nicht mehr aufgefangen werden.

Die Tatsache, dass Softwareentwicklung nur dann erfolgreich sein kann, wenn Organisation, Technologie und sozialer Kontext zueinander passen, lässt sich den Studierenden nur schwer vermitteln. Am besten geeignet sind dabei längere, eigenverantwortlich durchgeführte Projekte, wie beispielsweise die oben genannten Projektgruppen. Um diesen Lernerfolg zu erzielen, bedarf es neben der Erstellung einer konkreten Software der begleitenden und kritischen Diskussion der gewählten Herangehensweise.

6. Schlussfolgerungen

Angesichts der Tatsache, dass ein erheblicher Teil der Informatikabsolventen in der industriellen Softwareentwicklung tätig wird, sollte ausgeschlossen werden, dass ein Informatikstudium absolviert werden kann, ohne dass grundlegende Softwaretechnologie-Kenntnisse erworben werden.

Projektgruppen oder vergleichbare Lehrveranstaltungen sollten auf der Grundlage softwaretechnologsicher Kennnisse und Erfahrungen durchgeführt werden, da sie oftmals die einzige Chance sind, um systematische und wohlfundierte Entwicklungsprozesse einzuüben.