Anforderungen an den Unterricht der Hochschulen im Fach Software Engineering aus Sicht eines Softwarehauses

Ralf Kneuper*

1. Dezember 1993

1 Einführung

1.1 Bedeutung des Fachs Software Engineering im Unterricht der Hochschulen

F.L. Bauer (zitiert nach [Wal90, S. 284]) definiert Software Engineering wie folgt:

> *Software Engineering ist eine Disziplin, die mit ingenieurmäßigen Mitteln und ökonomischem Vorgehen dem Entwickler hilft, qualitativ hochwertige Software zu erstellen und zu pflegen.*

Software Engineering (SE) behandelt die allgemeinen Prinzipien der systematischen Erstellung von Software und ist daher ein zentrales Querschnittsthema für den gesamten Informatikbereich. Es umfaßt nicht nur fachliche Inhalte, sondern wesentlich auch eine (ingenieurmäßige) Denk- und Arbeitsweise bei der Software-Entwicklung.

Ein großer Teil der Informatik befaßt sich mit der Entwicklung von Software für spezifische Fragestellungen (z.B. Datenbanken, Compilerbau oder Künstliche Intelligenz). Daraus folgt, daß für diesen Teil der Informatik SE eine wesentliche Grundlage der Arbeit darstellt. Nach Ansicht des Autors sollte deshalb eine gründliche Ausbildung in SE (einschließlich einer Einführung in

*SOFTWARE AG, Neue Bergstraße 9-13, 64665 Alsbach-Hähnlein

die Teilgebiete Projektmanagement und Qualitätsmanagement) ein wesentlicher Teil der Ausbildung *aller* Informatikstudierenden sein (nicht nur als Wahlpflichtfach), der aber z.T. durchaus auch in andere Veranstaltungen, insbesondere andere Praktika, integriert sein kann (vgl. dazu Kap. 3.2). Insbesondere ist SE aus Sicht des Autors in der Informatikausbildung wichtiger als technische Einzelthemen wie Datenbanken, Künstliche Intelligenz oder theoretische Informatik.

Anforderung 1: *Unterricht im Fach SE ist notwendig für alle Informatiker und sollte Projektmanagement (nicht nur für Projektleiter) und Qualitätsmanagement einschließen.*

Die im folgenden beschriebenen Anforderungen an den Unterricht der Hochschulen im Fach SE (SEUH) sind jeweils *Mindestanforderungen*, die aus Sicht des Autors an die Ausbildung für alle Informatiker bestehen. Darüber hinaus sollte natürlich die Möglichkeit bestehen, zusätzlich SE oder Teilgebiete davon wahlweise wesentlich intensiver zu studieren.

Die hier verwendete Argumentation setzt voraus, daß das Ziel eines Studiums in erster Linie die Ausbildung für eine Berufstätigkeit, im Gegensatz zur Bildung und persönlichen Reifung, ist. Aus Sicht des Autors ist dies für das Informatikstudium und insbesondere die SE-Ausbildung im wesentlichen der Fall und wird deshalb hier als gegeben vorausgesetzt. Da die Studierenden nach Abschluß ihres Studiums aber Persönlichkeiten und keine Maschinenbediener sein sollten, sollte ein Studium daneben auf jeden Fall auch einen Anteil an persönlichkeitsbildenden Themen enthalten (z.B. in Form von Veranstaltungen über soziale Verantwortung oder als Studium Generale),

Die Bedeutung von kommunikativen Fähigkeiten für die Arbeit von Software-Entwicklern und damit auch für den SEUH wird z.B. deutlich aus einer Untersuchung von Brodbeck et. al. [BS*93]:

> *Fassen wir 'typische' SE-Tätigkeiten mit 'unterstützenden' SE-Tätigkeiten zusammen, so ist mit 55% der Gesamtarbeitszeit ein leichtes Übergewicht bei den Tätigkeiten festzustellen, die vorwiegend technische Fachqualifikation erfordern. Sitzungen, Beratung und Gespräche sowie Projektorganisation und Präsentation machen ca. 31% der Gesamtarbeitszeit aus. D.h. bei etwa einem Drittel der Arbeitszeit werden Arbeitsanforderungen an Mitarbeiter in SE-Projekten gestellt, die nicht nur fachliche Kompetenzen sondern insbesondere auch Kompetenzen für Kommunikation und Kooperation voraussetzen.*

Anforderung 2: *Am Ende der SE-Ausbildung sollten die Studierenden nicht nur mit dem Computer kommunizieren können, sondern auch kommunikative und soziale Kompetenz besitzen.*

1.2 Einige Grundbegriffe der Didaktik

Das Ziel der Vermittlung von Lehrstoff kann sein, daß die Studierenden ihn anschließend *kennen*, *wissen* oder *können*.

Kennen besagt, daß die Studierenden den Lehrstoff in Erinnerung haben und die wesentlichen Aspekte auch wiedergeben können. Dies kann im Rahmen von Vortrag oder Vorlesung erreicht werden.

Wissen besagt, daß die Studierenden den Lehrstoff nicht nur kennen, sondern ihn auch einordnen können, Zusammenhänge verstanden haben und z.B. beurteilen können, wann und wie er sinnvoll umgesetzt werden kann. Um dies zu erreichen, reichen Vortrag oder Vorlesung nicht aus, sondern man benötigt Lehrformen, bei denen die Studierenden selbst aktiv mitarbeiten, z.B. Diskussionen und kleinere Übungen.

Können schließlich besagt, daß die Studierenden den Lehrstoff selbständig in der Praxis umsetzen können. Dazu ist es unbedingt notwendig, daß sie das im Rahmen einer Übung, eines Praktikums oder ähnlichem schon einmal selbst gemacht haben.

Im folgenden wird deshalb nach diesen drei Kategorien unterschieden. Wird dort gefordert, daß die Studierenden einen bestimmten Lehrstoff wissen oder können sollten, so folgt daraus insbesondere, daß er nicht nur im Rahmen von Vorlesungen vermittelt werden darf.

Eine andere Gliederung unterscheidet zwischen kognitiven, affektiven und psychomotorischen Lernzielen. Während psychomotorische Lernziele für den SEUH nahezu irrelevant sind, sollten affektive, also die Einstellung der Studierenden zum Lehrstoff betreffende, Lernziele im Unterricht deutlich stärker berücksichtigt werden, als das jetzt der Fall ist. Ein derartiges affektives Lernziel wäre z.B. die Förderung des "egoless programming" [Wei71].

1.3 Der Autor

Eine Arbeit wie die vorliegende ist notwendigerweise subjektiv und durch die Erfahrungen des Autors geprägt. Aus diesem Grund soll er hier kurz vorgestellt werden. Nach dem Studium der Mathematik arbeitete der Autor in einem Forschungsprojekt im Bereich der formalen Methoden zur Software-Entwicklung und promovierte mit einer Arbeit aus diesem Bereich. Seit etwa vier Jahren arbeitet er in der Qualitätsmanagementgruppe der SOFTWARE AG und koordinierte dort u.a. die Entwicklung eines internen Seminars zum Thema "Qualitätsmanagement bei der Software-Entwicklung".

Die SOFTWARE AG wurde 1969 gegründet und zählt international zu den führenden unabhängigen Systemsoftware-Anbietern und Dienstleistern der Informationsverarbeitung. Das Unternehmen ist in über 60 Ländern vertreten und der größte Software-Exporteur Deutschlands.

Parallel zu seiner Arbeit dort hatte der Autor auch Lehraufträge an der TH Darmstadt im Bereich Software Engineering/Qualitätssicherung und betreute Praktikumsgruppen und Diplomarbeiten.

Der Autor dankt seinen Kollegen R. Hüttenberger, K. Münch, S. Tost und R. Weiß für viele Anregungen und Ideen, die in diese Arbeit eingeflossen sind. Die hier beschriebenen Meinungen sind jedoch die des Autors und stimmen nicht unbedingt mit der Meinung dieser oder anderer Kollegen überein. Insbesondere handelt es sich bei diesem Artikel nicht um eine offizielle Stellungnahme der SOFTWARE AG.

2 Ziele des Unterrichtes im Fach Software Engineering

Dieses Kapitel beschreibt Ziele des SEUH in Form von Anforderungen (in Fortsetzung der Anforderungen 1 und 2). Aus diesen Anforderungen werden dann im folgenden Kapitel 3 Schlußfolgerungen für Gestaltung und Inhalte des SEUH abgeleitet.

2.1 Prozeßorientierung

Ein zentrales Ziel des SEUH sollte aus Sicht des Autors sein, daß die Studierenden um die Prozeßorientierung des Software Engineering wissen und Verständnis für die Gesamtzusammenhänge in einem Software-Entwicklungsprojekt haben. Ohba [Ohb93, S. 30] formuliert diesen Aspekt wie folgt:

> *In software production, process technology (i.e. know-how) is the key element. Understanding basic theories of computer science (e.g., algorithms) and of software engineering (e.g., generic processes and techniques) is necessary, but insufficient to produce high-quality products or services (e.g., system integration) and to be competitive. Process technology, which can only be learned through the experience of producing software or services, is what drives a company's success.*

Studierende sollten am Ende ihrer SE-Ausbildung wissen, daß gute Methoden und Werkzeuge noch kein gutes Ergebnis garantieren, sondern daß die

Organisation[1] sowie Motivation und soziale Kompetenz der Projektmitarbeiter mindestens genauso viel Einfluß haben. Wenn Projekte scheitern, dann ist die Ursache sehr häufig in diesem Bereich zu finden und nur selten im rein technischen Bereich — auch wenn es auf den ersten Blick nicht immer so aussieht. Die Durchführung eines Projektes umfaßt sowohl SE-Aktivitäten als auch Aktivitäten wie Marketing, Vertrieb oder Druck der Dokumentation. Auch die SE-Aktivitäten selbst umfassen ein breites Spektrum vom Einsatz eines geeigneten Phasenmodells und einer geeigneten Entwicklungsmethode über das Konfigurationsmanagement und Prüfungen im Projektverlauf, z.B. in Form von Reviews und Tests, bis hin zu Installations- und Auslieferungsverfahren.

Erst das *Zusammenwirken* all dieser einzelnen Aktivitäten führt zu erfolgreichen Projekten. Sind einzelne Glieder dieser Kette zu schwach, so wird auch die gesamte Kette schwach bleiben. Sind gar Kettenglieder nicht verbunden oder fehlen völlig (z.B. Konfigurationsmanagement oder Zusammenarbeit im Team), so ist die Stärke der übrigen Glieder (z.B. Entwicklungsmethoden) für den Gesamterfolg irrelevant.

Diese Tatsache wird anschaulich dargestellt durch die Ungleichung

$$\boxed{O > M > T}$$

Organisation hat größeren Einfluß auf die Qualität der Ergebnisse als die eingesetzten *Methoden*, und diese wiederum haben größeren Einfluß als die verwendeten Werkzeuge (*Tools*). Insbesondere können Werkzeuge eine große Hilfe bei der Entwicklung sein und viel Verwaltungsarbeit abnehmen, aber sie ersetzen nicht die eigentliche kreative Denkarbeit bei der Entwicklung.

Diese Tatsache ist vielen Hochschulabsolventen nicht bewußt, sondern kommt häufig erst nach einigen Jahren (manchmal sehr teuer erkaufter) Erfahrung. Inwieweit dieses Bewußtsein bei den SE-Dozenten an den Hochschulen immer vorhanden ist, sei an dieser Stelle dahingestellt.[2]

Anforderung 3: *SEUH sollte diesen Lernprozeß unterstützen und beschleunigen und den Studierenden das Verständnis für Software-Entwicklung als komplexem Prozeß, an dem Menschen beteiligt sind und nicht nur Maschinen, vermitteln.*

---

[1] Der Begriff *Organisation* wird hier im Sinne von *Ablauforganisation* verwendet, also der Regelung der Abläufe (Prozesse) in einem Unternehmen. Die Aufbauorganisation, also die Regelung von Zuständigkeiten und Verantwortlichkeiten, hat dagegen bei der Software-Entwicklung vergleichsweise geringen Einfluß auf die Ergebnisse.

[2] Auch der Autor selbst hatte nach Abschluß seines Studiums große Illusionen über den Einfluß "besserer", sprich formaler, Entwicklungsmethoden auf die Qualität der fertigen Softwareprodukte. Es dauerte einige Zeit, bis ihm deutlich wurde, daß diese nur ein Einflußfaktor unter vielen sind.

2.2 Ingenieurmäßige Denk- und Arbeitsweise

Ein direkt aus der in Abschnitt 1.1 zitierten Definition des SE ableitbares Ziel des SEUH ist die Vermittlung einer ingenieurmäßigen und ökonomischen Denk- und Arbeitsweise. Dazu gehört z.B. das Verständnis von Programmieren als Problemlösung, nicht als Selbstzweck. Für den SEUH folgt daraus insbesondere die Betonung von Problemanalyse und Entwurf sowie von systematischer statt trickreicher Programmierung.

Anforderung 4: *SEUH sollte den Studierenden eine ingenieurmäßige und ökonomische Denk- und Arbeitsweise vermitteln.*

Ein weiteres Kennzeichen der zu vermittelnden ingenieurmäßigen Denk- und Arbeitsweise ist die permanente Verbindung zwischen Theorie und Praxis und daraus abgeleitet ein 'pragmatischer' Umgang mit der vermittelten Theorie. Baber [Bab92, S. 5] beschreibt dies mit den Worten

> *Theorie und Praxis ergänzen sich. Weder die eine noch die andere reicht aus. Auf eine ausgewogene Kombination der beiden kommt es bei einer ingenieurwissenschaftlichen Aufgabe oder Tätigkeit an.*

Dies ist beispielsweise dann gefordert, wenn Probleme in einem Projekt auftreten. Einerseits schreibt Humphrey über Organisationen mit geringer 'Prozeßreife' zu Recht [Hum88, S. 75]

> *... The best test is to observe how such an organization behaves in a crisis. If it abandons established procedures and reverts to merely coding and testing, it is likely to be at the Initial Process level. After all, if the techniques and methods are appropriate, they must be used in a crisis and if they are not appropriate, they should not be used at all.*

Andererseits dient der Einsatz einer Methode oder einer Vorgehensweise nicht als Selbstzweck, sondern der Qualität der Ergebnisse und ihrer effizienten Erstellung. Stellt sich im Einzelfall heraus, daß der Einsatz nicht (mehr) diesen Zielen dient, dann muß die Methode oder Vorgehensweise angepaßt oder ihr Einsatz sogar ganz eingestellt werden. Die Fähigkeit, in solchen Fällen einen sinnvollen Mittelweg zu finden, ist sicher sehr schwierig im SEUH zu vermitteln, stellt aber trotzdem ein wichtiges Ziel dar.

Anforderung 5: *Nach Abschluß der SEUH-Ausbildung sollten die Studierenden in der Lage sein abzuwägen, wann Methoden und Verfahrensweisen (langfristig wirkende) Mittel zum Zweck sind und wann sie zum Selbstzweck ausarten.*

2.3 Affektive Lernziele

Neben den oben genannten *kognitiven* gibt es eine Reihe von *affektiven* Lernzielen, die sich also auf die Einstellung der Studierenden zu bestimmten Inhalten beziehen, nicht auf die Vermittlung der Inhalte selbst. Wichtigstes affektives Ziel des SEUH ist sicher das folgende:

Anforderung 6: *Nach Abschluß der SEUH-Ausbildung sollten die Studierenden nicht nur die Inhalte des SE beherrschen, sondern auch deren Bedeutung und Nützlichkeit erkennen und eine positive Einstellung dazu haben, sie also nicht nur als (manchmal) notwendiges Übel betrachten.*

Ist diese positive Einstellung nicht vorhanden, dann werden die vermittelten Inhalte nicht freiwillig, sondern nur unter Druck angewandt, und die Ergebnisse sind entsprechend schlechter.

Ein weiteres wichtiges Ziel des SEUH sollte die Unterstützung des "egoless programming" im Sinne von [Wei71] sein.

Anforderung 7: *Nach Abschluß der SEUH-Ausbildung sollten die Studierenden Programme (und auch andere Dokumente wie z.B. Spezifikationen) nicht als ihr persönliches Werk verstehen, das es gegen Kritik von anderen (z.B. Reviewern oder Testern) zu verteidigen gilt, sondern als Ergebnis einer Teamarbeit, deren Ziel ein möglichst gutes Gesamtergebnis ist.*

In diesem Sinne sind Reviews oder Inspektionen selbstverständlicher Teil der Software-Entwicklung und eine Hilfe für den Entwickler bzw. Autor. Derzeit treffen diese Verfahren häufig noch auf Akzeptanzprobleme, weil sie zuerst als Eingriff in die Privatsphäre des Entwicklers verstanden werden.

2.4 Vermittlung von Fachwissen

Neben den bisher beschriebenen Anforderungen, die sich vor allem auf die Einstellung und die Denk- und Arbeitsweise der Studierenden beziehen, muß der SEUH natürlich auch das zugehörige Fachwissen vermitteln.

Anforderung 8: *SEUH sollte das für die ingenieurmäßige Erstellung von Software notwendige Fachwissen vermitteln.*

Da die Kluft zwischen den zu vermittelnden theoretischen Grundlagen und der industriellen Praxis teilweise noch sehr groß ist, ist es wichtig, daß die Studierenden die relevanten Stoffe wirklich beherrschen (im Sinne von Wissen

oder Können), damit sie diese Inhalte auch nach Abschluß des Studiums in der Praxis vorantreiben können. Deshalb sollten im Idealfall die Inhalte jeweils etwas über die praktisch eingesetzten Methoden etc. hinausgehen, aber nicht zu weit. Forschungsorientierte Themen sind dagegen für die Mehrheit der Studierenden wenig hilfreich und sollten daher nur als mögliche Vertiefungsgebiete angeboten werden.

3 Schlußfolgerungen für den Unterricht der Hochschulen im Fach Software Engineering

In diesem Kapitel werden einige Schlußfolgerungen für den SEUH, die sich aus den gestellten Anforderungen ergeben, behandelt. Dabei ist zu berücksichtigen, daß sich die geforderten Lernziele nur selten getrennt vermitteln lassen. Besonders deutlich wird dies bei den affektiven Lernzielen — eine eigene Unterrichtseinheit zur Vermittlung einer positiven Einstellung zum SE (Anforderung 6) wäre beispielsweise sicher nicht sinnvoll.

3.1 Praktika

Um die beschriebenen Ziele zu erreichen, ist eine Mischung aus Theorie und Praxis notwendig. Diese Einsicht ist weitverbreitet — offen bleibt die konkrete Gestaltung dieser Mischung. Sicher gehören dazu eine Vorlesung o.ä. zur Vermittlung der theoretischen Grundlagen sowie eine größere Praktikumsaufgabe während des Studiums[3], auch wenn diese nicht ausreichen, alle oben beschriebenen Ziele umzusetzen.

Um das Prozeßverständnis der Studierenden zu fördern (vgl. Anforderung 3), sollten im Rahmen des Praktikums möglichst vollständige Projekte durchgeführt werden — im Idealfall sogar über mehrere Semester verteilt mit sich wandelnden Anforderungen, um die Wartung von Software einzubeziehen und auf diese Weise auch die nur langfristig erkennbaren Vorteile von systematischer Entwicklung und Dokumentation der Ergebnisse sichtbar zu machen (vgl. Anforderung 6).

Dabei ist eine gemeinsame Betreuung der Praktika durch Hochschule und Industrie anzustreben, um auf diese Weise die unterschiedlichen Schwerpunkte zu

[3] An der TH Darmstadt z.B. wird ein derartiges SE-Praktikum angeboten, bei dem eine Gruppe von üblicherweise sechs Studierende über ein Jahr verteilt ein (oft von Firmen gestelltes) Projekt durchführt. Dabei wird ein Aufwand von ca. sieben Wochen Vollzeit pro Studierendem angesetzt.

verbinden. Empfehlenswert ist auch eine abschließende Auswertung des Praktikums, um gemachte Erfahrungen auszuwerten und aus ihnen zu lernen. Dabei sollten vor allem die Vor- und Nachteile der verwendeten Methoden und Vorgehensweisen analysiert werden (vgl. Anforderung 5).

3.2 Integration mit anderen Fächern

Ein erhebliches Problem, das sich aus den gestellten Anforderungen ergibt, ist der dafür benötigte Zeitaufwand. Werden alle gestellten Anforderungen getrennt von der übrigen Informatikausbildung verfolgt, so füllt allein der SEUH einen erheblichen Teil des Studiums. Einige Anforderungen, vor allem Anforderungen 4 und 6, sind auf diesem Weg sogar fast nicht zu erreichen.

Ein Lösungsansatz für diese Probleme ist die Integration des SEUH in die übrige Informatikausbildung, z.B. durch Durchführung möglichst aller Praktika, Entwicklungsaufgaben, etc. nach den Grundsätzen des SE (mit Projektplan etc.). Dies setzt eine intensive Zusammenarbeit innerhalb der Hochschule voraus, außerdem müssen dazu *alle* Dozenten das Arbeiten nach SE-Prinzipien selbst beherrschen und umsetzen. Dies stellt ein erstrebenswertes Ziel dar, auch wenn es bisher auf breiter Ebene wohl weder an der Hochschule noch in der Wirtschaft erreicht wird.

Eine derartige Integration stellt also sehr hohe Anforderungen an die Hochschulen. Andererseits wird dadurch die Vermittlung der entsprechenden Denkweise bei der Entwicklung von Software erheblich unterstützt (insbesondere Anforderungen 4 und teilweise 3 und 6). SE ist dann kein getrenntes Thema, das z.B. mit der Entwicklung eines Compilers oder eines DBMS nichts zu tun hat, sondern integraler Bestandteil jeder Software-Entwicklung.

Dabei sollten auch Projektmanagement und Qualitätsmanagement als Teilgebiete des SE mit einbezogen werden, indem z.B. grundsätzlich ein Projektplan erstellt (und seine Einhaltung verfolgt) wird und alle wesentlichen Ergebnisse einem Review unterzogen werden (vgl. Anforderungen 1 und 7).

3.3 Unterrichtsinhalte

Bei der Festlegung der Inhalte des SEUH sind in erster Linie existierende Standards in der Breite wichtig, neueste Entwicklungen dagegen weniger (z.B. formale Methoden). Der Schwerpunkt sollte also auf dem *Stand der Technik* liegen, nicht auf dem *Stand der Wissenschaft*, dabei aber auch aktuelle Trends mit berücksichtigen.

3.3.1 Entwicklungsmethoden

Eines der im SEUH traditionell am intensivsten behandelten Themen sind die verschiedenen Entwicklungsmethoden. Dabei sollte ein breites Spektrum von Methoden und Verfahren behandelt, nicht aber einzelne (z.B. Objektorientierung) als Allheilmittel dargestellt werden, das alle Probleme löst. Bei den einzelnen Methoden sollten jeweils Vor- und Nachteile behandelt und deutlich werden, daß im einzelnen Projekt jeweils neu entschieden werden muß, welche Methoden und Verfahren im Einzelfall angemessen sind und wie diese angepaßt werden können oder müssen (vgl. Anforderung 5). Die Gefahr ist hier groß, einen einmal für richtig gehaltenen Ansatz unbedingt weiterzuverfolgen, ohne die damit ursprünglich verfolgten Ziele im Auge zu behalten.

Der Einsatz von Werkzeugen für die verschiedenen Entwicklungsaufgaben, insbesondere von CASE-Werkzeugen, sollte dabei nach Möglichkeit auch behandelt werden, aber erst, nachdem die entsprechenden Methoden jeweils "mit Papier und Bleistift" beherrscht werden.

3.3.2 Andere Inhalte

Andere wesentliche Inhalte, die ebenfalls im SEUH behandelt werden sollten, sind u.a.

- Vorgehenskonzepte, z.B. das Vorgehenskonzept des Bundes oder das Projektmodell einer Firma wie der Software AG (vgl. Anforderung 3)
- relevante Normen wie z.B. die ISO 9000-Reihe oder die SE-Standards der IEEE — hier genügt Kennen
- Dynamic Systems Modelling (vgl. Anforderung 3) — hier genügt ebenfalls Kennen
- Konfigurationsmanagement und Versionsverwaltung
- Grundlagen des Projektmanagements und des Qualitätsmanagements.

4 Zusammenfassung

Zusammenfassend läßt sich sagen, daß Ziel des SEUH in erster Linie sein sollte, das Grundverständnis für die Probleme des SE zu fördern. Dazu gehört insbesondere das Verständnis von Software-Entwicklung als (technischer und sozialer) Prozeß. Die Inhalte sollten sich stärker am Stand der Technik als am Stand der Wissenschaft orientieren, dafür aber zu erheblichen Teilen auf der Stufe des

Wissens oder Könnens vermittelt werden, um die oft recht große Kluft zwischen Theorie und Praxis überwinden zu können.

Um dies zu erreichen, ist eine intensive Zusammenarbeit zwischen Hochschule und Industrie notwendig (z.B. durch Vorträge, Lehraufträge, Praktika, Studien- und Diplomarbeiten). Hier sind beide Seiten gefordert, den dafür notwendigen Aufwand auch wirklich zu betreiben. Angenehmer Nebeneffekt einer solchen Zusammenarbeit ist einerseits der verbesserte Technologie-Transfer von der Hochschule in die Praxis, andererseits das bessere Verständnis der Hochschullehrer für die Probleme der Praxis. Zu fordern ist hier auch eine ausreichende praktische Erfahrung der Dozenten außerhalb des Hochschulbereiches (zumindest der Dozenten für SE, besser aber der Dozenten aller Teilgebiete der Informatik), wobei diese praktischen Erfahrungen möglichst nicht zu lange in der Vergangenheit liegen sollten, da die entsprechenden Eindrücke sonst doch deutlich verblassen.

Daneben ist auch eine verbesserte Zusammenarbeit innerhalb der Hochschulen notwendig, um die Arbeitsweisen des SE in alle Software-Entwicklungsaufgaben zu integrieren.

Literaturverzeichnis

[Bab92] Robert L. Baber. Konzepte für die Erstellung möglichst fehlerfreier Software in der Vergangenheit und Zukunft. *HMD — Theorie und Praxis der Wirtschaftsinformatik*, (163):3–16, Januar 1992.

[BS*93] F.C. Brodbeck, S. Sonnentag, T. Heinbokel, W. Stolte und M. Frese. Tätigkeitsschwerpunkte und Qualifikationsanforderungen in der Software-Entwicklung: Eine empirische Untersuchung. *Softwaretechnik-Trends*, 2(13):31–40, Mai 1993.

[Hum88] Watts S. Humphrey. Characterizing the software process: A maturity framework. *IEEE Software*, 5(2):73–79, März 1988.

[Ohb93] Mitsuru Ohba. The impact of organizational knowledge on quality. *American Programmer*, 6(6):28–39, Juni 1993.

[Wal90] Ernest Wallmüller. *Software-Qualitätssicherung in der Praxis*. Carl Hanser Verlag, München, 1990.

[Wei71] Gerald M. Weinberg. *The Psychology of Computer Programming*. Van Nostrand Reinhold, 1971.