Teamentwicklung in studentischen Projekten

Doris Schmedding, TU Dortmund

Doris.schmedding@tu-dortmund.de

Zusammenfassung

Teamfähigkeit ist eine der Anforderungen an Software-Entwickler, die sehr häufig in Stellenanzeigen genannt wird. Unter Teamfähigkeit wird bei Wikipedia die Fähigkeit verstanden, mit anderen zusammen sozial zu agieren und sich und sein Können im Sinne einer Gruppenaufgabe optimal einzubringen.

Studentische Software-Entwicklungsprojekte bieten die Möglichkeit durch Learning-by-doing neben den technischen Fähigkeiten auch die Softskills der Studierenden auf dem Gebiet der Teamarbeit zu trainieren. Dieser Artikel stellt in einem Software-Praktikum erfolgreich erprobte Methoden vor, mit denen Lehrende ohne großen Aufwand die Teamentwicklung in studentischen Projekten unterstützen und die Reflexion der Erfahrungen anleiten können, um daraus neue Impulse für eine bessere Zusammenarbeit zu gewinnen.

Einleitung

Das Software-Praktikum an der TU Dortmund (https://sopra.cs.tu-dortmund.de/wiki/) ist eine langjährig etablierte Lehrveranstaltung, die neben der Anwendung von Methoden und Verfahren aus der Software-Technik in Software-Entwicklungsprojekten darauf abzielt, die Teamfähigkeit der Studierenden zu verbessern. In Gruppen zu je 8 Studierenden werden unter Anleitung eines erfahrenen Tutors nacheinander zwei Software-Projekte durchgeführt. Gleichzeitig nehmen 5-12 Gruppen am Praktikum teil. Das Praktikum hat laut Prüfungsordnung einen Umfang von 4 SWS. Konkret heißt das, dass sich eine Gruppe an zwei Terminen pro Woche im Seminarraum trifft, um am Projekt zu arbeiten. Daneben arbeiten die Studieren auch zuhause oder im Rechnerpool an der Universität an dem Projekt.

In einem Projekt sind technische, fachliche, organisatorische und soziale Probleme zu lösen. Projekte scheitern weniger an mangelndem technischen oder fachlichem Wissen, vielmehr stellt die Lösung von organisatorischen und sozialen Problemen eine größere Herausforderung dar (Fleischmann u.a., 2005). Da ich diese Beobachtung aus eigener Erfahrung nur bestätigen kann, möchte ich die Studierenden ebenso, wie wir die Einarbeitung in bis dahin noch unbekannte Werkzeuge wie SVN oder einen GUI-Builder durch speziell vorbereitete Tutorials unterstützen, auch auf die bis dahin noch unbekannte Arbeit in einem Projektteam vorbereiten. Für das Lernen ist es wichtig, die eigenen Erfahrungen mit den neuen Arbeitstechniken zu reflektieren und bei Bedarf die Vorgehensweisen zu verändern. Nur so lassen sich aus den Erfahrungen im ersten Projekt Konsequenzen für das zweite Projekt ziehen. In (Lewerentz u. Rust, 2001) wird ausführlich auf die Bedeutung der Reflexion der Erfahrungen in der Projektarbeit eingegangen. Den Vorschlag der Autoren, neben einem gemeinsamen Reflexionsbericht zusätzlich am Ende jedes Teilprojekts persönliche Reflexionsberichte von allen Studierenden einzufordern, halte ich für wenig praktikabel. Zumindest Dortmunder Informatik-Studierende möchten lieber Java-Code als Selbstreflexionen schreiben.

In diesem Artikel gebe ich zunächst eine Einführung in die Teamentwicklung. Dann stelle ich zwei Beispiele für Übungen zur Teambildung vor, mit denen ohne großen Aufwand die Teambildung unterstützt werden kann. Zu einer derartigen Übung gehört auch die Reflexion der Erfahrungen. Das Thema Reflexion wird vertieft, indem eine Methode vorgestellt wird, die zur Halbzeit des Praktikums eingesetzt wird.

Teamentwicklung

Ein Team unterscheidet sich von einer Gruppe durch die Tatsache, dass die Mitglieder eines Teams gemeinsam eine Aufgabe lösen und/oder ein gemeinsames Ziel verfolgen. In diesem Sinne können wir also bei einem studentischen Software-Entwicklungsprojekt mit mehreren Studierenden von einem Team ausgehen.

Die Entwicklung von einer zufällig zusammengestellten Gruppe von Studierenden mit heterogenem Vorwissen zu einem gut funktionierenden Team läuft in einer Abfolge von Phasen ab (siehe Abb. 1), die in der Literatur in Anlehnung an die Teamuhr von Tuckman (Tuckman, 1965) gerne mit

den eingängigen Begriffen *Forming, Storming, Norming* und *Performing* bezeichnet werden.

Diese Phasen der Teamentwicklung werden bis auf die Forming-Phase bei neuen Aufgaben oder Zielen, oder wenn ein neues Mitglied zur Gruppe hinzukommt, wieder neu durchlaufen. Anstelle des Forming tritt dann die Reorganisation des Teams, das *Reforming* ein (Vigenschow u.a., 2009).

In (Marks, 2009) wird nicht nur erläutert, was in den einzelnen Phasen unter den Gruppenmitgliedern passiert, sondern es werden auch die Rolle und die Aufgaben der Lehrenden erläutert. Nachfolgend werden die einzelnen Phasen der Teamentwicklung vorgestellt, wie ich sie als Betreuerin von studentischen Software-Entwicklungsgruppen oft beobachten konnte.

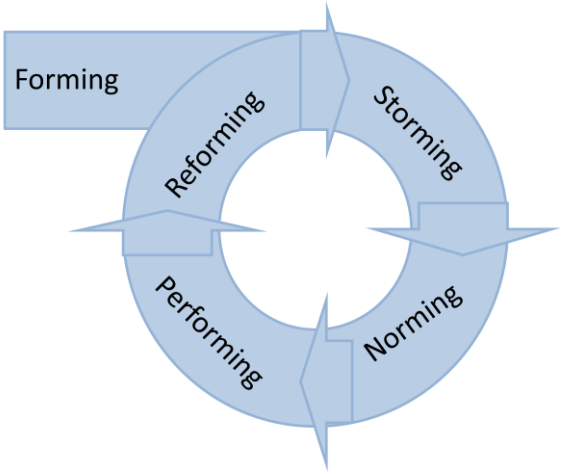

Abb.1: Phasen der Teamentwicklung

Anfangsphase (Forming)

Nach der Gruppenbildung befindet sich die Gruppe zunächst in der Anfangsphase, die von großer Unsicherheit geprägt ist. Die Aufgabenstellung ist noch unbekannt. Die Mitglieder der Gruppe kennen die Persönlichkeiten und Vorkenntnisse der anderen Gruppenmitglieder noch nicht. Der Leitung der Gruppe werden Autorität und soziale Fähigkeiten unterstellt, die diese nutzen kann, um die Gruppe über die schwierige Anfangssituation hinweg zu führen.

Konfliktphase (Storming)

Nach der Phase des Kennenlernens fühlen sich die Gruppenmitglieder sicher. Es folgt eine Phase der Auseinandersetzung um die Einflussmöglichkeiten in der Gruppe. Oberflächlich geht es dabei oft um Sachthemen. Viele Konflikte finden aber auf der Beziehungsebene statt, auf der es um Sympathie und Antipathie, um die eigenen und die fremden Werte und Einstellungen und um das Ansehen in der Gruppe geht, das auf vermuteter fachlicher Kompetenz und Vertrauen beruht. Das Eisbergmodell der Kommunikation (siehe Abb.2) lässt sich auf die Kommunikation in Gruppen übertragen (Virgenschow u.a., 2009) und hilft die Zusammenhänge und die Ursachen für langwierige Diskussionen zu verstehen.

Die Gruppe arbeitet an einer gemeinsamen Zieldefinition und Normen; Verhaltensregeln für die Gruppe werden festgelegt. Als Ergebnis der Konfliktphase entsteht eine informelle Hierarchie, in auch die Lehrenden einbezogen sind. Die Reaktion der Lehrenden, die die Ziele der Lehrveranstaltung vertreten müssen, wird ausgetestet.

Normierungsphase (Norming)

Die Normierungsphase ist geprägt durch die Entwicklung eines Wir-Gefühls im Projektteam. Nachdem ein praktikables Regelwerk zur Lösung von Konflikten erarbeitet worden ist, können Konflikte relativ reibungsfrei beigelegt werden. Wir beobachten, dass die Gruppe sich um die Integration von Außenseitern bemüht und gegenüber Außenstehenden abschließt. Dazu wird oft auch die Gruppenleitung gezählt. Insgesamt wächst die Arbeitsfähigkeit der Gruppe in dieser Phase stark an.

Arbeitsphase (Performing)

In der letzten Phase, der Arbeitsphase (Performing), fließt die gesamte Teamenergie in die Aufgabenbewältigung. Wegen des hohen Gruppenzusammenhalts sind nun auch Spitzenleistungen möglich. Die Arbeit wird innerhalb des Teams nach Effizienzgesichtspunkten verteilt und erledigt. Die Lehrenden benötigen nur noch wenig Aufwand, da sich die Gruppe weitgehend selbst organisiert und kontrolliert. Allerdings müssen Lehrende jetzt darauf achten, dass die Lehr- und Lernziele nicht aus den Augen verloren werden. In meiner Lehrveranstaltung gilt die Regel, dass jeder alle Aktivitäten zumindest ansatzweise einmal ausgeführt hat, was nicht unbedingt effizient im Sinne des Projektteams ist.

Projekt 0 - Übung zur Teambildung

Wenn das Zusammenwachsen einer Gruppe zu einem gut funktionierenden und produktiven Team so schwierig ist, stellt sich die Frage, wie Lehrende eine Gruppe bei der Teamentwicklung unterstützen können. In (Fleischmann, 2005) wird ein recht umfangreiches dreitägiges Teamtraining beschrieben, für das in unserem vierstündigen und einsemestrigen Praktikum leider keine Zeit ist. Auch mit geringerem Aufwand können die Lehrenden durch gezielte Übungen und den Anstoß zur Reflexion Einfluss auf die Teamentwicklung nehmen, wenn sie sich über die Abläufe und ihre eigene Rolle im Entwicklungsprozess der Gruppe bewusst sind.

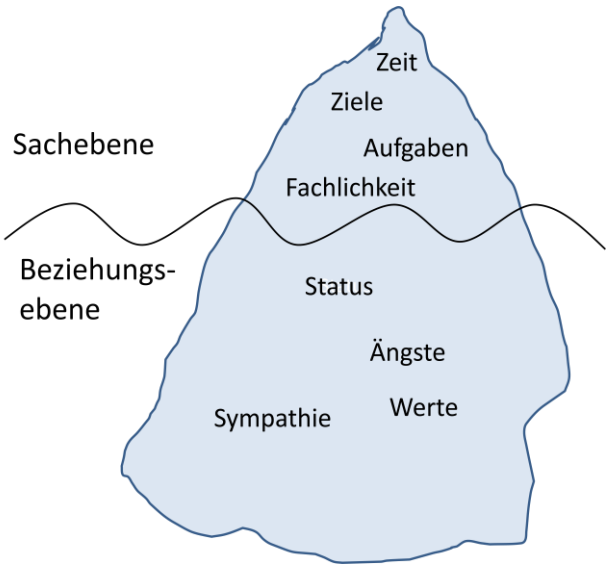

Abb. 2: Eisbergmodell der Kommunikation

In der Anfangsphase können Übungen zur Teambildung helfen, das Kennenlernen zu erleichtern und die Abläufe im Team zu veranschaulichen. Besonders geeignet für Software-Entwicklungsteams sind Übungen, die einen Projektcharakter haben und an denen sich die Analogie zu Software-Projekten gut zeigen lässt. Derartige Übungen bezeichne ich deshalb mit Projekt 0.

Turmbau

Sehr bewährt haben sich in meiner Arbeit im Software-Praktikum Bastel-Projekte wie der Turmbau als Einstieg in die Projektarbeit.

Aufgabenstellung: Aus einer vorgegebenen Anzahl von Blättern Papier soll z.B. ein möglichst hoher Turm gebaut werden. Zwei Studierende aus der Gruppe beobachten das Bauteam und berichten über ihre Beobachtungen. Die restlichen 6 Gruppenmitglieder bilden das Bau-Team. Eine Schere wird als Werkzeug zur Verfügung gestellt. Der Turm soll in 30 Minuten gebaut werden.

Als Beobachter kann man sehr schön die Phasen der Teamentwicklung verfolgen. Anfangs betrachten die Studierenden recht ratlos die Blätter und die Schere. Dann stehen meist konkurrierende Vorschläge von einzelnen Gruppenmitgliedern im Raum. Je nach Sympathie oder Vertrauen in Kompetenz der Vorschlagenden schließen sich einzelne Gruppenmitglieder einer Idee an. Dann wird entweder sehr lange diskutiert oder relativ schnell ein Regelwerk gefunden, um sich auf einen Lösungsvorschlag zu einigen. In der Bauphase sind die meisten aktiv, manche halten sich zurück. Es wird geschnitten, gefaltet, gerollt und zusammengesteckt. Am Ende ist die Gruppe meist mit ihrem Werk sehr zufrieden.

Durch das praktische Tun lockert sich schnell die Atmosphäre in der Gruppe. Ein Einstiegsprojekt bietet die Möglichkeit, die Persönlichkeit der Mitglieder kennen zu lernen. Entscheidungsprozesse können eingeübt werden.

Bevor man die Beobachter berichten lässt, sollten die Mitglieder des Bauteams nach ihren Erfahrungen befragt werden. Folgende Fragen bieten einen guten Einstieg in die Diskussion über den Problemlösungsprozess mit den Studierenden:

- Welche Verhaltensweisen haben der Gruppe bei der Lösung der Aufgabe geholfen?
- Welche Verhaltensweisen haben die Gruppe bei der Lösung der Aufgabe behindert?
- Wer hat sich am meisten beteiligt?
- Wer hat sich am meisten zurückgehalten?
- Wie habt Ihr die Diskussionen und den ganzen Lösungsprozess erlebt?
- Was hat mir/der Gruppe die Übung gebracht?
- Wie ist die Gruppe mit der vorgegebenen Zeit zurechtgekommen?

Die Rolle der Beobachter ist für die Reflexion des Geschehens wichtig. Als nicht unmittelbar am Bau Beteiligte spiegeln sie der Gruppe ihr Verhalten und ihre Vorgehensweise.

Die durchaus gewollte Analogie zum Software-Projekt mit *Planung, Entwurf* und *Realisierung* wird von den Studierenden durchaus gesehen. Man kann über Abweichungen der Realisierung von der Planung und dem Entwurf diskutieren. Auch die Themen *Zeitmanagement* und *knappe Ressourcen* werden in diesem Kurzprojekt sehr gut veranschaulicht.

Die Aufgabenstellung lässt sich durch erweiterte Anforderungen und andere Themengebiete leicht variieren. Das Konzept der *Modularisierung* lässt sich z.B. durch den Bau eine Brücke bestehend aus Pfeilern und einer Auflage aufgreifen. *Schnittstellen* lassen sich z.B. bei einer Kugelbahn gut integrieren, indem man explizit eine Steckverbindung zwischen getrennt zu entwickelnde Komponenten fordert. Die zusammengesetzte Kugelbahn eignet sich auch, um das Thema unabhängiges *Testen von Komponenten* in das Einführungsprojekt mit aufzunehmen.

Am Ende der Analyse der Übung sollte die Frage diskutiert werden, welche Konsequenzen die Erfahrungen aus Projekt 0 für das vor der Gruppe liegende Software-Entwicklungsprojekt haben sollen. Typischen Schlussfolgerungen sind, dass die Diskussion strukturierter erfolgen sollte und dass eine Sitzungsmoderation sinnvoll wäre.

Sin-Obelisk

Eine andere Art von Einstiegsaufgaben stellen gemeinsam von der Gruppe zu lösende Rätsel dar, mit denen man die Bedeutung des Informationsaustausches und die Kooperation üben kann. Ein Beispiel für diesen Typ von Aufgaben ist der sogenannte Sin-Obelisk

(http://www.spielekiste.de/archiv/diverses/komm/komm_004.shtml).

Auf vielen kleinen Kärtchen (>30) stehen jeweils Informationen geschrieben, die die Gruppe zur Beantwortung einer Frage (Lösung einer Aufgabe) benötigt. Eingestreut sind nutzlose Informationen und Fragen. Jedes Gruppenmitglied bekommt eine Reihe von Informationskärtchen. Nur durch Zusammentragen der relevanten Informationen und geschickte Aneinanderreihung gelingt es der Gruppe, die Eingangsfrage zu beantworten. Im Fall des Sin-Obelisken ist es die Frage nach dem Wochentag, an dem er fertig gestellt wird. Eine der benötigten Informationen ist z.B. die Höhe des Obelisken.

Für die gesamte Übung einschließlich Reflexion reicht eine Stunde völlig aus. Nach meiner Erfahrung findet eine Gruppe von Informatikstudierenden in etwa 10-15 Minuten die Lösung.

Wie beim Turmbau werden Beobachter bestimmt, die der Gruppe ihre Beobachtungen vortragen. Mit etwa den gleichen Fragen zum Problemlösungsprozess wie beim Turmbau kann die Diskussion mit den Studierenden initiiert werden. Offene Fragen, die reihum abgefragt werden, helfen, die Studierenden zu aktivieren. Durch die Fokussierung auf bestimmte Punkte kann die Aufmerksamkeit der Beobachter gelenkt werden.

Ein Ziel der Übung ist es, den Umgang mit verstreuter Information im Problemlösungsprozess zu trainieren. Die Analogie zur Situation in einem neu zusammengestellten Team für ein Software-Entwicklungsprojekt besteht z.B. in den unterschiedlichen Vorkenntnissen, die gemeinsam zur Lösung der Aufgabe genutzt werden. Daneben kann man Kooperationsbereitschaft und Führungsverhalten und den Umgang mit Konflikten bei der Problemlösung in der Gruppe studieren. Auch lassen sich sehr schön die Phasen der Teamentwicklung beobachten, von Unsicherheit über Erarbeitung eines Regelwerks zum gut funktionierenden Interagieren.

Konstruktive Reflexion der Erfahrungen

Wenn Lehrende in studentischen Software-Entwicklungsprojekten das Ziel „Stärkung der Teamfähigkeit“ ernst nehmen und mehr erreichen wollen als nur eine Lockerung der Atmosphäre und eine Stärkung des Selbstbewusstseins durch eine schöne und schnelle Lösung, muss bei den Studierenden eine eigene Reflexion der Erfahrungen in Gang gesetzt werden. Dazu ist zumindest ein kleiner Einblick in die Theorie der Gruppendynamik notwendig, den ich in der Einführungsveranstaltung zum Software-Praktikum gebe.

Bei der Thematisierung gruppendynamischer Prozesse geht man davon aus, dass neben der Sachebene, die sich mit den Aufgaben und Zielen der Lehrveranstaltung, den Anforderungen und den technischen Fragen im Projekt beschäftigt, auch die Beziehungsebene existiert, die die Sympathie der Gruppenmitglieder untereinander, ihre Werte und Normen und ihre Ängste beinhaltet (vgl. Abb.2). In der Diskussion von Teamprozessen mit Betroffenen setzen wir uns als Lehrende der Software-Technik dem Risiko aus, schwerwiegende Probleme Einzelner und in den Beziehungen untereinander an die Oberfläche zu befördern. Die sorgfältige Behandlung derartiger Probleme braucht sehr viel Zeit und sollte ausgebildeten Trainern überlassen werden (Vigenschow u.a., 2009). Nicht-Psychologen kommen deshalb bei der Beschäftigung mit der Gruppendynamik, der Prozessanalyse und dem Rollenverhalten schnell an ihre Grenzen.

Beim Einholen von Kritik an den Abläufen in der Gruppe und an der Lehrveranstaltung und damit auch an der eigenen Rolle als Lehrende sollte man sich darüber klar sein, dass derartige Rückmeldungen auch für einen selbst unangenehm sein können. Daher sollte zunächst auf gewisse Spielregeln, sogenannte Feed-Back-Regeln, hingewiesen werden (Vigenschow u. Schneider, 2007). So haben wir die Chance, mit Kritik konstruktiv umzugehen.

Da im Software-Praktikum an der TU Dortmund von den Studierenden zwei Software-Entwicklungsprojekte durchgeführt werden, bietet es sich an, nach dem ersten Projekt der Erfahrungen zu reflektieren. Als Leiterin der Lehrveranstaltung besuche ich alle Gruppen, um mit den Studierenden anhand der folgenden Fragen ihre Erfahrungen zu diskutieren:

- Was ist im 1. Projekt gut gelaufen?
- Was muss im 2. Projekt besser werden?

Die Antworten lasse ich auf Karten notieren und clustern. Zugelassen sind sowohl Kritik an der Technik, dem Lehrkonzept als auch an der Arbeit im Team. Die Karten bieten den Vorteil, dass auch die Stillen zu Wort kommen. Aus Häufungen lässt die besondere Relevanz eines Punktes ablesen.

In der Regel wird das Klima in der eigenen Gruppe als kooperativ erlebt und auch in dieser Runde genannt. Die gegenseitige Beteuerung der guten Teamarbeit verstärkt noch einmal das gute Klima. Allerdings werden Schwächen in der Zusammenarbeit durchaus auch gesehen und thematisiert. Daraus können selbst formulierte Regeln für ein geändertes Arbeitsverhalten abgeleitet werden. Beispielsweise soll beim Auftreten von Problemen den anderen früher Bescheid gegeben werden, damit sie schneller unterstützend eingreifen können. Oder beim Einchecken in das Repository soll immer ein Kommentar geschrieben werden, damit die anderen wissen, was geändert wurde. Ein der-

artiges selbst erarbeitetes Regelwerk ist bei der Kooperation im Team äußerst hilfreich.

Auch als Verantwortliche für das Praktikum kann ich aus diesen Sitzungen viel mitnehmen. Das sind Verbesserungsvorschläge für die technische Ausstattung, die eingesetzten Werkzeuge, aber auch für das Konzept der Veranstaltungen, z.B. Anregungen wie die Einarbeitung in Tools besser unterstützt werden kann.

Ein derartiges Reflexionsgespräch mit einer Gruppe dauert etwa eine Stunde.

Fazit

In einem Software-Entwicklungspraktikum stehen naturgemäß die Inhalte im Vordergrund, die sich „direkt" mit der Software-Entwicklung beschäftigen. Auf diesem Gebiet sind wir als Software-Techniker Fachleute und das wollen die Studierenden von uns lernen. Dennoch wissen wir aus langjähriger Berufserfahrung und Beobachtung der studentischen Teams, welche Bedeutung die Zusammenarbeit der Gruppenmitglieder für den Erfolg oder Misserfolg eines Projekts hat.

Neben den bereits zitierten Büchern von Vigenschow und anderen, die sich mit den Softskills einmal aus der Sicht der Entwickler (Vigenschow u. Schneider, 2007) und einmal aus der Sicht eines Projektleiters (Vigenschow u.a., 2009) beschäftigen, verdeutlicht der lesenswerte Ratgeber von Hedwig Kellner (Kellner, 2006), dass neben der fachlichen Kompetenz auch die Teamfähigkeit sehr wichtig für eine Karriere in der Software-Entwicklung ist. Arbeitgeber erwarten von Mitarbeitern neben fachlicher Kompetenz auch soziale Kompetenz. In komplexen Projekten werden Teamworker gebraucht, die ihre guten Einzelleistungen zu Gesamtleistungen kombinieren.

Ganz konkret wird in (Hölzle u. Grünig, 2006) gezeigt, dass für ein erfolgreiches Projektmanagement soziale Sensibilität benötigt wird. Anhand des Eisbergmodells (Abb.2) wird verdeutlicht, dass sich die wirklichen Gründe für Ressourcenprobleme "unter der Wasseroberfläche" einer guten Planung verbergen, die in der Regel nur einen kleinen Anteil am Gesamterfolg eines erfolgreichen Projektmanagements beiträgt.

Wenn also soziale Kompetenz für das Gelingen von Projekten so wichtig ist, sollte ihre Bedeutung auch in den Lehrveranstaltungen thematisiert werden. Für das Themengebiet *Teamarbeit* eignen sich insbesondere studentische Software-Entwicklungsprojekte, da sich hier Theorie und Praxis gut kombinieren lassen. Die Studierenden können sich selbst und die anderen Teammitglieder beobachten und ihr neues Wissen direkt ausprobieren.

In (Fleischmann u.a., 2005) und in (Lewerentz u. Rust, 2001) werden zwei Beispiele vorgestellt, wie im Rahmen von Software-Praktika mit relativ großem zeitlichem und personellem Aufwand erfolgreich daran gearbeitet wird, die Softskills der Studierenden zu verbessern. Ich habe hier gezeigt, dass auch mit weitaus weniger aufwendigen, gut platzierten Maßnahmen die Teamarbeit unterstützt werden kann.

Die vorsichtige Integration einiger Inhalte in eine bestehende und etablierte Veranstaltung, wie ein Kurzvortrag über Teamarbeit, ein Teamentwicklungsprojekt und die angeleitete Reflexion der Erfahrungen, ist einfach zu realisieren, kostet fast keine zusätzliche Zeit, lohnt sich aber auf jeden Fall, denn diese Investition wird durch ein besseres Arbeitsklima belohnt und verbessert die Vorbereitung der Studiereden auf den Beruf.

Literatur

Fleischmann, A., Spies, K., Neumeyer, K. (2005): Teamtraining für Software-Ingenieure. In: Löhr, K.-P., Lichter, H. (Hsg.): SEUH 9, 26-40.

Kellner, H. (2006): Soziale Kompetenz: Für Ingenieure, Informatiker und Naturwissenschaftler. Hanser.

Lewerentz, C., Rust, H. (2001): Die Rolle der Reflektion in Softwarepraktika. In: Lichter, H., Glinz, M. (Hsg.): SEUH 7, 73-86.

Marks, F. (2009): Gruppendynamik und Hochschulunterricht – Gruppendynamische Prozesse im Seminar. Erschienen in Berendt, B., Voss, P., Wildt, J., Tremp, P. (Hrsg.): Neues Handbuch Hochschullehre.

Tuckman, B.W. (1965): Developmental sequences in small groups. Psychological Bulletin, 63, 348-399.

Vigenschow, U., Schneider, B. (2007): Soft Skills für Softwareentwickler. dpunkt.verlag.

Vigenschow, U., Schneider, B., Meyrose, I. (2009): Soft Skills für IT-Führungskräfte und Projektleiter. dpunkt.verlag.

Hölzle, P., Grünig, C., (2006): Projektmanagement. Professionell führen - Erfolge präsentieren. Haufe.