# Verteilte Softwarepraktika mit der Projektmethode

Till Schümmer, Stephan Lukosch und Jörg M. Haake
Fachbereich Informatik
FernUniversität in Hagen
Universitätsstr. 1
58084 Hagen
{till.schuemmer, stephan.lukosch, joerg.haake}@fernuni-hagen.de

**Abstract:** Praktika sind ein wesentlicher Bestandteil in der akademischen Ingenieursausbildung. In diesem Artikel stellen wir einen *Blended Learning*-Ansatz für solche Praktika in der Informatik vor. Wir zeigen, wie die Projektmethode für verteilte Softwarepraktika verwendet werden kann, in deren Rahmen verteilte Gruppen eine Software realisieren. Zur Illustration beschreiben wir, wie unser Ansatz durch die kooperative Lernumgebung *CURE* unterstützt werden kann. Erfahrungen bei der Verwendung unseres Ansatzes in bisher zwei durchgeführten Praktika zeigen, dass Gruppen von diesem Ansatz profitieren.

## 1 Einleitung

Softwarepraktika sind ein wichtiger Bestandteil der Informatikausbildung an Universitäten. Sie vertiefen das Verständnis von Programmierkonzepten durch aktive Anwendung der Theorie auf ein realitätsnahes Problem. Durch Einbettung der Lernziele in einen anwendungsbezogenen Kontext wird die Erfahrung authentischer und im späteren Berufsleben leichter anwendbar. Anstatt zu lernen was getan werden muss, erfahren die Teilnehmer wie ein Problem gelöst wird.

Der Ansatz des problembasierten Lernens (PBL) [Woo94] liefert hierfür einen didaktischen Hintergrund: Studierende sollen gemeinsam ein realitätsnahes Problem lösen, das durch einen Dozenten gestellt wird. PBL wird zunehmend bei der Ingenieursausbildung eingesetzt [HE02]. Beim Einsatz von PBL in gruppenorientierten Softwarepraktika ergibt sich oft der folgende Ablauf:

- Der Dozent kündigt das Praktikum mit einer Aufgabenbeschreibung an.
- Studierende melden sich für das Praktikum an.
- Studierende bilden Gruppen und erste gruppenbezogene Normen [Tuc65].
- Studierende arbeiten gemäß einer vorgeschriebenen Methode zusammen (z.B. dem Unified Software Development Process [JBR99]).
- Studierende reichen Ergebnisse zu vorgeschriebenen Meilensteinen ein.

- Das Praktikum endet nach Abgabe mit der Bewertung des letzten Meilensteins durch den Dozenten.

Entsprechende Praktika in Deutschland sind z.B. das *eXtreme Programming-Labor* an der Universität Karlsruhe [BFD04], das *Software Engineering*-Praktikum an der Technischen Universität Darmstadt [SD98] oder das Datenbankpraktikum an der FernUniversität in Hagen [BBB+04]. Die ersten beiden Beispiele finden an Präsenzuniversitäten statt. Die Praktikumsgruppen treffen sich dort häufig und führen Entwurfs- oder Pair Programming-Sitzungen [WU01] durch. Die Teilnehmer wenden hierbei eine Entwurfsmethode an, die vor dem Praktikum in einer Vorlesung gelehrt wurde. Beim eXtreme Programming-Labor in Karlsruhe stellen Tutoren sicher, dass die Teilnehmer, die in der eXtreme Programming Methode [Bec99] vorgeschriebenen Rollen einhalten. Deswegen werden die meisten gemeinsamen Aufgaben am selben Ort (d.h. im Labor) bearbeitet.

Das Datenbankpraktikum ist dagegen ein Beispiel für ein verteiltes Praktikum. Es verwendet einen vergleichbaren Gruppenprozess, wobei die Teilnehmer in einer virtuellen Umgebung interagieren (*BSCW* [AM99] und *IRC*) und sich zu keinem Zeitpunkt physisch treffen. Die Dozenten definieren die Designmethode und die zu erstellenden Ergebnisse. Alle Gruppen bearbeiten dieselbe Aufgabe und können die Arbeit frei verteilen. Die Dozenten spielen die Kunden und beobachten den Gruppenprozess. Sie greifen nur ein, wenn die Gruppe den intendierten Prozess falsch anwendet.

Unsere Erfahrungen an der FernUniversität in Hagen zeigen mehrere Probleme bei der Anwendung dieses Ansatzes in der Fernlehre, bei der Studierende großflächig verteilt und oft in Vollzeit beschäftigt sind:

- Motivationsprobleme wenn z.B. die Arbeit wichtiger als das Praktikum ist,
- Koordinationsprobleme wenn z.B. parallele Lehrveranstaltungen das Einhalten mehrerer Zeitpläne erfordern,
- Prozessprobleme wenn die vorgeschriebene Entwurfsmethode als zu schwergewichtig für die Praktikumsaufgabe angesehen wird und
- Kommunikations- und Interaktionsprobleme wenn sich die Gruppe nicht gut genug kennt und deshalb Probleme bei der Gruppeninteraktion erlebt.

Zusammengenommen führen diese Probleme zu einer hohen Abbruchquote. Bei einem Gruppenpraktikum sind diese Abbrüche auch für die restlichen Gruppenmitglieder bedrohlich, da sie eine Umstrukturierung und erhöhte Arbeitslast bewirken.

In diesem Beitrag berichten wir über einen alternativen Ansatz für verteilte Softwarepraktika. Dieser basiert auf der Projektmethode und wird von der kollaborativen Lernumgebung CURE [HSB+04] unterstützt. Wir beginnen mit einem Überblick über die Projektmethode und stellen dann unseren Vorschlag für die Anwendung auf verteilte Software Engineering Praktika vor. Dann zeigen wir, wie CURE die Umsetzung dieses Ansatzes unterstützt. Abschließend berichten wir über den Einsatz der Methode in zwei Praktika und schließen mit Ideen für weitere Arbeiten.

## 2 Die Projektmethode für verteilte Software Engineering Praktika

Nach Knoll [Kno97] liegen die Wurzeln der heute in Europa oft genutzten Projektmethode in der Architekturausbildung des ausgehenden 16. Jahrhunderts. Der erste einflussreiche Aufsatz über die Projektmethode wurde vom amerikanischen Bildungstheoretiker Kilpatrick [Kil18] veröffentlicht, der den Begriff Projekt als eine gezielte Handlung im Kontext der Bildung für Kinder definierte. Das Hauptziel bestand darin, Kindern den Erwerb von Wissen in praktischen und sozialen Kontexten zu erlauben. Dies lässt sich auch auf die Erwachsenenbildung übertragen. Nach Kilpatrick sollen Projekte in vier Phasen durchgeführt werden:

- **Zielsetzung**: Studierende definieren die Projektidee und das Ziel,
- **Planung**: die zur Lösung des vorgeschlagenen Problems notwendigen Schritte werden identifiziert,
- **Ausführung**: die Schritte werden durchgeführt, und
- **Beurteilung**: Studierende beurteilen das Ergebnis des Prozesses und vergleichen es mit ihren ursprünglichen Projektzielen.

Es ist wichtig, dass alle Phasen durch die Studierenden und nicht durch den Lehrer ausgeführt werden. Der Lehrer hilft wo nötig, aber die Studierenden gestalten selbst die Ziele und Vorgehensweisen ihres Prozesses. Dies ist der wesentliche Unterschied zu dem in der Einleitung beschriebenen üblichen Verfahren.

Gudjons [Gud97] identifizierte acht wichtige Charakteristika für die Anwendung der Projektmethode. Demnach sollte ein Projekt

- sich an den Interessen der Beteiligten (Studierende und Dozenten) orientieren,
- durch die Studierenden selbstverantwortlich organisiert werden (mit Zustimmung der Dozenten),
- in der Gruppe einen zielgerichteten Planungsprozess verfolgen,
- interdisziplinär im Bezug auf die Lernziele sein,
- ein hochgradig sozialer Prozess sein, der die sozialen Kompetenzen aller Teilnehmer verbessert,
- eine gesellschaftliche Praxisrelevanz haben, d.h. das Ergebnis sollte wichtig für die Teilnehmer sein,
- von der Kooperation aller Teilnehmer beim gemeinsamen Handeln und nicht durch direkte Anweisungen geprägt sein und
- verschiedene Sinne aktivieren.

Obwohl diese Charakteristika einzeln auch in anderen didaktischen Ansätzen vorkommen, kombiniert die Projektmethode sie auf eine für kooperatives Lernen außerordentlich relevante Weise. Motivationsprobleme werden durch die Zielsetzungsphase adressiert, die sich auf selbst definierte Aufgaben, Zielorientierung und die persönliche Relevanz konzentriert. Koordinationsprobleme werden durch die Planungsphase angegangen: die Studierenden können iterativ planen und so sich ändernde Gruppenbedürfnisse berücksichtigen. Die Adäquatheit des Prozesses wird in der Planungs- und Ausführungsphase sichergestellt. Die Studierenden werden von den Dozenten zur fortwährenden Reflexion und Anpassung ihrer Arbeitsprozesse angehalten und lernen so, den Prozess zu verbessern.

Zur Anwendung der Projektmethode in einem verteilten Softwarepraktikum schlagen wir die folgende Sequenz von Aktionen vor, die, wie jeweils angegeben, verteilt oder am selben Ort durchgeführt werden:

| **Administration (verteilt, asynchron)** | |
|---|---|
| Ankündigung | Die Dozenten kündigen das Praktikum mit einer Beschreibung des in Projekten zu adressierenden Problemraums an. |
| Anmeldung | Studierende melden sich zum Praktikum unter Angabe ihres Hintergrundwissens an. |
| Auswahl | Die Dozenten wählen geeignete Teilnehmer aus. |
| **Zielsetzung (verteilt, asynchron)** | |
| Erzeugung der Projektidee | Die Dozenten teilen den Teilnehmern die Anforderungen für vorzuschlagende Projekte mit. Diese Anforderungen stellen sicher, dass vorgeschlagene Projekte alle Lernziele berücksichtigen. Die Teilnehmer erstellen eigene Projektvorschläge, die voneinander unterschiedlich sein sollen. |
| Diskussion der Projektidee | Die Teilnehmer kommentieren gegenseitig ihre Projektideen. Die Diskussion soll die Ideen verdeutlichen und ein gemeinsames Verständnis über mögliche Projekte ergeben. |
| Auswahl der Projektideen | Die Dozenten wählen die interessantesten Projektideen aus, die zu den Praktikumszielen passen. |
| Vorbereitung der Präsentationen | Die Autoren der ausgewählten Projektideen erstellen einen kurzen (Werbe-)Vortrag für die erste Präsenzphase. |
| **Gruppenbildung und Planung (gleicher Ort, synchron)** | |
| Gegenseitiges Kennenlernen | Das gegenseitige Kennenlernen (Socializing) ist wichtig für verteilte Teilnehmer, die sich sonst nie treffen. Die Teilnehmer müssen sich vor der Gruppenbildung kennen lernen. Spiele, die zur sozialen Struktur der Gruppe passen, können hierbei helfen [MKT98]. |
| Präsentation der Projektideen | Die ausgewählten Teilnehmer präsentieren die Projektideen in einer Plenumssitzung (Fragen sind nach jedem Vortrag möglich). |
| Gruppenbildung | Die Studierenden bilden Gruppen von 5 bis 7 Teilnehmern. Dabei stützen sie sich auf die Präferenzen für Projektideen und auf persönliche Qualifikationen (z.B. Erfahrung in Projektleitung, Netzwerkprogrammierung), die sie in ihrer Vorstellung während der Kennenlernphase beschrieben haben. So soll eine Gruppe mit einer ausgewogenen Kompetenzverteilung entstehen, in der mindestens ein Teilnehmer Interesse an der Projektleitung hat. |

| | |
|---|---|
| Projektauswahl | Die Gruppen diskutieren verschiedene Projektideen und erarbeiten Vorschläge für drei bevorzugte Ideen. Die Vorschläge müssen den gewählten Ansatz für das jeweilige Projekt skizzieren. Anhand der Vorschläge weisen die Dozenten dann jeder Gruppe eine Projektidee zu. |
| Einführung und Erarbeitung der Entwurfsmethode | Die Dozenten präsentieren ausgewählte Software Engineering Methoden, die den Teilnehmern als Ideen für die Erarbeitung des Vorgehens für ihr Projekt dienen. Dabei sind problemspezifische (z.B. Programmierrichtlinien) und soziale (z.B. Kommunikationsregeln) Fragestellungen zu berücksichtigen. |
| Erstellen des Arbeitsplans | Die Teilnehmer zerlegen das Problem in kleine Arbeitspakete, schätzen den Aufwand für diese, und bringen sie in eine Bearbeitungsreihenfolge (Workflow). |
| Rollenzuweisung | Die Teilnehmer übernehmen die Rollen, die sie bei der Festlegung ihres Gruppenprozesses identifiziert haben, und weisen sich Spezialgebiete und Arbeitspakete zu. Alle Rollen und Gebiete sollen von mind. zwei Teilnehmern abgedeckt werden, um Wissenstransfer zu unterstützen und die Konsequenzen von Ausfällen zu minimieren. |
| **Ausführung (verteilt, asynchron)** | |
| Gruppenarbeit | Die Teilnehmer bearbeiten die Arbeitspakete alleine oder in Kleingruppen (abh. von Kooperationsmöglichkeiten und Aufgabe). Resultate werden der Gruppe verfügbar gemacht und diskutiert. |
| Monitoring | Die Dozenten und Projektleiter beobachten den Arbeitsplan und stellen sicher, dass die Gruppe sich etwaiger Verzögerungen bewusst ist. Wenn Verzögerungen zunehmen, werden die Teilnehmer zur Anpassung ihres Zeitplans angehalten. Die Projektleiter müssen sicherstellen, dass die Teilnehmer ihre Rollen ausfüllen. Dozenten intervenieren nur, wenn die Projektleiterrolle nicht ausgefüllt ist oder der Projektleiter um Hilfe bittet. Die Gruppe kann Rollenzuweisungen ändern (aber nicht zu oft). |
| Gruppenübergreifender Austausch | Die Gruppen sollen regelmäßig ihre Erkenntnisse austauschen. Dies kann z.B. bei gleichartigen Fragestellungen zu gegenseitigem Lernen führen. |
| **Beurteilung (gleicher Ort, synchron)** | |
| Ergebnisvorstellung | Jede Gruppe hält eine Produktpräsentation. Die anderen Gruppen und die Dozenten geben Verbesserungshinweise. |
| Reflexion | In der Projekt-Retrospektive [Ker01] vergleichen die Teilnehmer die ursprünglichen Projektziele mit dem Ergebnis. Sie berichten über Teile der Projektarbeit, die gut funktionierten oder fehlschlugen (z.B. Aspekte der Kommunikation oder der Projektleitung). Die Dozenten weisen die Gruppe auf solche Aspekte hin. |
| Bewertung | Zusammen mit der Gruppe legen die Dozenten eine Leistungsbeurteilung fest (abh. vom Erfahrungsgewinn der Teilnehmer). |

## 3 Blended Learning in einem Praktikum: Unterstützung in CURE

Die kooperative Lernplattform CURE (Collaborative Universal Remote Education) unterstützt verschiedene kooperative Lernszenarien [HSB+04]. Im nächsten Abschnitt beschreiben wir kurz die Hauptkonzepte von CURE und zeigen wie CURE den Einsatz der Projektmethode unterstützt.

### 3.1 CURE in Kürze

CURE basiert auf der Metapher der virtuellen Räume [GR03], die häufig zur Strukturierung von Kooperation eingesetzt wird. Hierbei dient ein Raum der Repräsentation eines virtuellen Orts für die Zusammenarbeit. In CURE können Räume verschiedene Dinge enthalten: Seiten (Inhalte), Kommunikationskanäle (z.B. Chat, Diskussionsforen) und Benutzer (siehe Abb. 1). Benutzer, die gleichzeitig im selben Raum sind, können miteinander mittels eines Chat synchron kommunizieren, der automatisch zwischen allen gleichzeitigen Benutzern des Raums etabliert wird. Sie können auch auf alle Seiten im Raum zugreifen. Veränderungen dieser Seiten sind für alle Benutzer im Raum sichtbar. Zur Definition von Zugriffsrechten auf Räumen wird das Konzept des virtuellen Schlüssels benutzt. Jeder Schlüssel definiert die Rechte des Inhabers auf dem zugehörigen Raum (z.B. Rechte zum Betreten des Raums, zum Erzeugen eines Unterraums, zum Editieren von Seiten oder zur Kommunikation im Raum). Räume mit einem öffentlichen Schlüssel sind für alle registrierten Benutzer von CURE zugreifbar.

Benutzer können einen Raum betreten, auf seine Kommunikationskanäle zugreifen und an der Zusammenarbeit im Raum teilnehmen. Ebenso können Benutzer Seiten erzeugen und editieren. Die Seiten können entweder direkt per einfacher WIKI-Syntax [LC01] editiert werden oder sie können binäre Dokumente oder Artefakte enthalten. Die WIKI-Syntax unterstützt Links zu anderen Seiten, Räumen, externen URLs und E-Mail-Adressen. Alle Artefakte werden durch den CURE-Server für den gemeinsamen Zugriff gespeichert. So bleiben bei Verlassen eines Raums seine Inhalte erhalten und Benutzer können später zurückkommen und weiterarbeiten.

Abb. 1 zeigt einen typischen Raum in CURE (die Nummern beziehen sich auf Details die im folgenden Abschnitt erläutert werden). Er enthält Dokumente (in Abb. 1 liest der Benutzer *Alexander* das Dokument *Homepage* im Raum *GoGo-Gadget* - ①), die von Benutzern mit ausreichenden Editierrechten manipuliert werden können ②. Der Raum stellt zwei Kommunikationskanäle zur Verfügung: eine *Threaded-Mailbox* ③ und einen Chat ④. Benutzer können die raumbasierte Mailbox zum Senden von E-Mails an den Raum verwenden, die dann von Benutzern mit Kommunikationsrechten für diesen Raum empfangen werden.

Durch miteinander verbundene Räume können strukturierte Lernumgebungen erstellt werden. Mittels eines Plenarraums kann die Kommunikation und die gemeinsame Bearbeitung von Dokumenten in einem Kurs (Vorlesung) oder einer ganzen

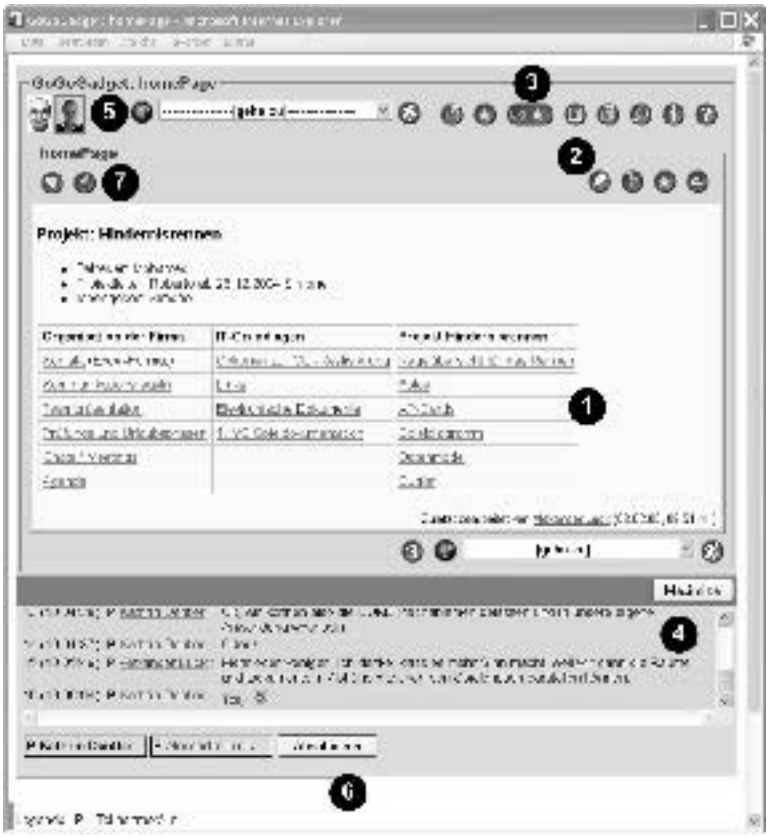


Abbildung 1: Ein Raum in CURE

Organisation unterstützt werden. Durch das Erzeugen neuer Räume für Untergruppen und Verbinden dieser Räume mit dem Kurs- oder Organisationsraum kann die gemeinsame Arbeit und Zusammenarbeit strukturiert werden. Verschiedene Typen von Awareness-Information in CURE unterstützen die Koordination zwischen den Benutzern eines Raums. Erstens können die Benutzer in den Raumeigenschaften sehen, wer Zugriff auf den Raum hat. Zweitens können Benutzer sehen, wer gerade im Raum anwesend ist ⑤. Drittens können die Benutzer bei eingeschaltetem Chat sofort miteinander reden ⑥. Viertens sorgt ein automatisch verschickter Mail-Report dafür, dass alle Benutzer des Raums erfahren, welche Änderungen im Raum seit dem letzten Report geschehen sind. Wenn Benutzer Seiten ändern, wird der alte Zustand als Vorversion gespeichert ⑦. Um den Zugriff auf die Änderungshistorie zu ermöglichen, speichert CURE die Versionen aller Seiten.

### 3.2 Unterstützung der Projektmethode für ein verteiltes Praktikum in CURE

In Abschnitt 2 haben wir fünf Phasen für die Durchführung eines Softwarepraktikums anhand der Projektmethode vorgestellt. Im Folgenden beschreiben wir kurz, wie CURE diese Phasen unterstützt. In den Präsenzphasen wurden alle Teilnehmer mit Notebooks ausgestattet, die per WLAN miteinander verbunden waren.

**Administration.** In CURE kann jeder Fachbereich der Universität einen öffentlichen Raum für seine Kurse anlegen. Für ein neues Praktikum richten die Dozenten einen neuen Praktikumsraum ein. Durch den von CURE automatisch an alle Benutzer verschickten Mail-Report erfahren die Studierenden vom neuen Raum (d.h. Praktikum). Im Report ist auch ein Link auf den Praktikumsraum enthalten. Mit Hilfe des Links können die Studierenden mehr über mögliche Themen im Praktikum erfahren. Sie können sich durch Beantragen eines Schlüssels für den Praktikumsraum zum Praktikum anmelden und dabei ihre Erfahrungen und Qualifikationen beschreiben. CURE informiert die Dozenten über die Schlüsselbeantragungen, die bis zum Erreichen der maximalen Teilnehmerzahl in Abhängigkeit von den dokumentierten Qualifikationen Schlüssel zuteilen.

**Zielsetzung.** Im Praktikumsraum finden die Teilnehmer Seiten mit zusätzlichen Informationen, z.B. die erste Aufgabe (Erstellen einer Projektskizze, die den dokumentierten Anforderungen genügt) und den Abgabetermin. Dazu müssen die Teilnehmer im Praktikumsraum eine Seite mit einer Skizze ihrer Projektidee erstellen. Nach der Abgabefrist diskutieren die Teilnehmer die Skizzen in der Mailbox oder dem Chat des Raums. Die Dozenten wählen die Skizzen aus, die den Lernzielen des Praktikums am besten entsprechen und kündigen diese im Praktikumsraum an. Die betroffenen Teilnehmer bereiten daraufhin ihre Präsentationen vor.

**Gruppenbildung und Planung.** In einem ersten Schritt der Präsenzphase stellen die Teilnehmer ihre Projektskizzen mit einer kurzen Präsentation vor. Zur Gruppenbildung erstellen alle Teilnehmer danach persönliche Seiten in CURE, auf denen sie ihre persönlichen Stärken und Interessen beschreiben (vergleichbar zum "*self image game*"[MKT98, S. 26]). Anhand dieser Seiten bilden die Studierenden selbstständig Gruppen um die Teilnehmer, die am Projektmanagement interessiert sind. Nachdem die Gruppen durch die Dozenten bestätigt wurde, erzeugt jede Gruppe einen eigenen Unterraum in CURE als gemeinsamen Arbeitsraum.

Als erste gemeinsame Aufgabe wählt jede Gruppe, die für sie interessantesten Projektskizzen. Für diese erstellt jede Gruppe in ihrem Arbeitsraum Seiten mit ausführlichen Projektbewerbungen. Anhand dieser Bewerbungen wählen die Dozenten dann ein Projekt für die Gruppe aus.

In Anschluss an die Projektauswahl stellen die Dozenten unterschiedliche Entwicklungsmethoden vor. Jede Gruppe wählt eine Entwicklungsmethode für die Ausführungsphase aus und stellt Regeln für die Nutzung von CURE während der Ausführungsphase auf (z.B. jeder liest CURE Mails mindestens jeden zweiten Tag, regelmäßige Treffen im CURE-Chat des Arbeitsraums). Als nächstes erzeugt jede Gruppe Seiten mit Beschreibungen der notwendigen Arbeitspakete und bringt diese auf

einer Index-Seite in eine chronologische Ordnung. Zusätzlich zu den Arbeitspaketen werden die einzelnen Verantwortlichkeiten und Rollen in CURE dokumentiert. Damit endet die erste Präsenzphase und es geht mit verteilter Zusammenarbeit weiter.

**Ausführung.** Die Gruppenmitglieder tragen sich als Bearbeiter für die einzelnen Arbeitspakete ein. Während der Arbeit nutzen sie CURE als Logbuch. Die Logbuchseiten und die von CURE automatisch an alle Benutzer verschickten Mail-Reports werden von den Projektleitern und den Dozenten zur Gruppenbeobachtung genutzt. Dozenten und Teilnehmer diskutieren die Meilensteine in der threaded Mailbox des Praktikumsraums. Wenn eine Gruppe den Anforderungen nicht genügt verlangen die Dozenten Abhilfe. Zur Förderung der Kommunikation zwischen Gruppen erzeugen die Dozenten themenzentrierte Unterräume im Praktikumsraum und fordern die Teilnehmer zur Diskussion übergreifender Themen auf.

**Beurteilung.** In der abschließenden Präsenzphase erzeugt jede Gruppe einen öffentlichen Unterraum zur Präsentation und Diskussion der Gruppenergebnisse. Das dort eingestellte Material dient als Grundlage für die Diskussion der Ergebnisse mit den anderen Gruppen in der Plenarsitzung. Danach reflektiert jede Gruppe über ihren Prozess (auf der Basis der Logs und anderer Artefakte, z.B. Seiten, Mails, Chat-Logs). Die Erkenntnisse aus der Reflexion werden als separate Seiten im Gruppenraum gespeichert. Die Teilnehmer können so auf sie zurückgreifen. Die Abschlussphase endet mit einer Gesamtbeurteilung der Gruppe.

## 4 Erfahrungen

Die beschriebene Methode und technische Unterstützung wurde in zwei Praktika im Fachbereich Informatik eingesetzt und evaluiert. Im ersten Praktikum (WS 03/04) entwickelten 34 Teilnehmer in 6 Gruppen kooperative Spiele. Im zweiten Praktikum (WS 04/05) entwickelten 21 Teilnehmer in 3 Gruppen Werkzeuge zur Unterstützung des kooperativen Lernens. In beiden Praktika verwendeten wir die eXtreme Programming Methode (XP) [Bec99].

### 4.1 Methode

**Setting.** Die Studierenden arbeiteten von zu Hause oder vom Büro aus und stehen mittels CURE über das Internet in Kontakt. Programmquellen wurden mittels CVS ausgetauscht. Der ersten Distanzphase folgte die erste Präsenzphase (4 Tage), gefolgt von ca. viermonatiger verteilter Zusammenarbeit bis zur Abschlusspräsenzphase (2 Tage). Während der verteilten Zusammenarbeit kommunizierten Dozenten mit Studenten mittels CURE und gelegentlicher Telefonate.

**Design.** Wir beobachteten Systemnutzung und die Interaktion.

**Probanden.** Teilnehmer waren reguläre Studierende, die zumeist arbeiteten und in Teilzeit studierten. Dozenten waren Professoren oder Mitarbeiter mit Erfahrung in E-Learning über das Internet sowie mit Expertenwissen über CURE.

**Vorgehen.** Die Dozenten entwickelten die Lernumgebung (Räume, Material) selbst. Die Studierenden erhielten kein Training in CURE. Sie wurden auf das online Manual verwiesen, das neben einem Tutorial auch ein Referenzhandbuch enthielt.

**Messgrößen und Evaluationsinfrastruktur.** Wir führten regelmäßige Interviews mit Dozenten und einigen Gruppen (während der Präsenzphasen) durch. Wir analysierten die von den Gruppen erzeugten Räume und Artefakte (inkl. Mailbox und Chat Logs). Dabei konzentrierten wir uns auf die vier identifizierten Probleme.

### 4.2 Resultate

**Motivationsprobleme.** Alle Teilnehmer erstellten gut ausgearbeitete Projektskizzen. Die Dozenten wählten für die Lernziele förderliche Skizzen aus. Die Beteiligung an der Diskussion während der Zielsetzungsphase blieb unzureichend. Jede Gruppe erstellte jeweils drei gute Projektbeschreibungen, was zeigt, dass sich die Gruppen mit den Skizzen identifizieren konnten. Es zeigten sich keine Motivationsprobleme. Dieser Trend setzte sich in der Planungs- und Ausführungsphase fort.

**Koordinationsprobleme.** Alle Gruppen erstellten konkrete Arbeitspläne (wie von den Dozenten verlangt). Die Gruppen verständigten sich auf Meilensteine und dokumentierten diese in CURE. Falsche Aufwandsabschätzungen oder persönliche Koordinationsprobleme führten zur Anpassung von Meilensteinen. Alle Gruppen pflegten den Projektplan.

**Prozessprobleme.** Alle Gruppen konstruierten eine an verteilte Teams angepasste Variante der XP Methode. Hauptaugenmerk lag auf dem Planungsprozess von XP. Dies führte dazu dass alle Gruppen ein tiefes Verständnis ihrer Aufgaben erreichten und sich deshalb auf sehr konkrete Arbeitspläne einigen konnten. Der resultierende Arbeitsplan wurde in CURE verwaltet.

**Kommunikations- und Interaktionsprobleme.** Der Gruppenbildungsprozess führte zu einem klaren Rollenverständnis und zu verantwortungsbewusster Projektleitung. Die Gruppen reflektierten oft über Rollenzuweisungen, die manchmal angepasst wurden (z.B. wegen krankheitsbedingtem Ausfall oder mangelnder Erfahrung). In allen Fällen waren sich die Gruppen über die Arbeitsverteilung im Klaren und zur Anpassung fähig. Alle Gruppen etablierten und dokumentierten Kommunikationsregeln, die auch tatsächlich befolgt wurden. Die Kommunikation war sehr aktiv und wies oft einen sozialen Anteil auf. Wir führen dies auf das explizite Phase zum gegenseitigen Kennenlernen und die Gruppenbildung während der ersten Präsenzphase zurück.

**Inhaltliche und soziale Lernziele.** Alle Gruppen produzierten funktionierende Ergebnisse und lernten viel über den Prozess der verteilten Softwareentwicklung.

Sie eigneten sich sowohl soziale als auch technische Fertigkeiten der rechnerunterstützten Projektarbeit an.

## 5 Diskussion und Ausblick

Die Analyse klassischer Präsenzpraktika zeigte mehrere Probleme für den Einsatz der Projektmethode in einer verteilten Lernumgebung: Motivationsprobleme, Koordinationsprobleme, Prozessprobleme und Kommunikations- und Interaktionsprobleme. Die Umsetzung der vorgestellten Kombination von Blended Learning mit der Projektmethode mit Unterstützung durch CURE ist nach unseren Erfahrungen ein erfolgreicher Weg, um verteilte Softwarepraktika zu veranstalten und die genannten Probleme zu lösen bzw. abzumildern. Die Ergebnisse zweier Praktika, die mit diesem Ansatzes durchgeführt wurden, zeigen, dass Dozenten und Studierende den Ansatz erfolgreich anwenden können, die genannten Probleme reduziert wurden und dass nicht nur Fähigkeiten zur Softwareentwicklung sondern auch soziale Fertigkeiten erlernt wurden. Motivationsprobleme konnten nicht beobachtet werden.

Bisher wurde die Projektmethode vor allem in reinen Präsenzveranstaltungen eingesetzt. Uns sind nur wenige Fälle von voll verteilten Praktika bekannt (z.B. [Tho02]). Die Kombination mit Blended Learning ist unserer Kenntnis nach neu. Die vorgeschlagenen Änderungen der Projektmethode, die gezeigte Unterstützung durch eine kooperative Lernumgebung und die Erfahrungsberichte stellen deshalb neue Einsichten für projektorientiertes kooperatives Lernen dar.

Unsere Erfahrungen haben gezeigt, dass problemorientierte Interaktion in einer verteilten Umgebung durchgeführt werden kann. Die explizite Phase zum gegenseitigen Kennenlernen, zur Gruppenbildung und die initiale Planung der Gruppenarbeit sollte aber in Präsenzphasen durchgeführt werden. Die ideale Aufteilung auf die Phasen und die perfekte Abstimmung zwischen sozialen Prozessen und technologischer Unterstützung bleiben aber offene Fragen. Darüber hinaus untersuchen wir zurzeit wie Groupware-Werkzeuge das gegenseitige Kennenlernen und die Gruppenbildung noch vor der ersten Präsenzphase unterstützen können.

## Literatur


[AM99] W. Appelt und P. Mambrey. Experiences with the BSCW Shared Workspace System as the Backbone of a Virtual Learning Environment for Students. In *Proceedings of ED-MEDIA99*, 1999.

[BBB+04] D. Becking, T. Berkel, S. Betermieux, M. Clossen, B. Feldmann, G. Rademacher und G. Schlageter. Motivation and Structuring in a Virtual Database Practical by Means of Roleplaying. In *Proceedings ICDE 2004*, 2004.

[Bec99] K. Beck. *eXtreme Programming Explained.* Addison Wesley, 1999.

[BFD04] C. Bunse, R. L. Feldmann und J. Dorr. Agile Methods in Software Engineering Education. In *Proceedings of XP 2004*, Juni 2004.

[GR03] Saul Greenberg und Mark Roseman. Using a Room Metaphor to Ease Transitions in Groupware. In M. Ackermann, V. Pipek und V. Wulf, Hrsg., *Sharing Expertise: Beyond Knowledge Management*, Seiten 203–256. MIT Press, Cambridge, MA, USA, Januar 2003.

[Gud97] Herbert Gudjons. *Handlungsorientiert lehren und lernen. Schüleraktivierung, Selbsttätigkeit, Projektarbeit.* Erziehen und Unterrichten in der Schule. Klinkhardt, Bad Heilbronn, Deutschland, 2.. Auflage, 1997.

[HE02] H. A. Hadim und S. K. Esche. Enhancing the engineering curriculum through project-based learning. In *32nd ASEE/IEEE Frontiers in Education Conference*, Boston, MA, 2002. IEEE Press.

[HSB+04] J. M. Haake, T. Schümmer, M. Bourimi, B. Landgraf und A. Haake. CURE – Eine Umgebung für selbstorganisiertes Gruppenlernen. *i-com Zeitschrift für interaktive und kooperative Medien*, September 2004.

[JBR99] Ivar Jacobson, Grady Booch und James Rumbaugh. *Unified Software Development Process.* Addison-Wesley, 1999.

[Ker01] N. L. Kerth. *Project Retrospectives: A Handbook for Team Reviews.* Dorset House Publishing Company, Incorporated, 2001.

[Kil18] W. H. Kilpatrick. The project method. *Teachers College Record*, 19:319–335, 1918.

[Kno97] M. Knoll. The project method: Its Vocational Education Origin and International Development. *Journal of Industrial Teacher Education*, 34(3), 1997.

[LC01] Bo Leuf und Ward Cunningham. *The Wiki Way.* Addison Wesley, 2001.

[MKT98] Solas M. Mc Kee, N. und H. Tillmann, Hrsg. *Games and Exercises – a manual for facilitators and trainers involved in participatory group events.* UNICEF-ESARO, 1998.

[SD98] Brunner M. Schroeder, U. und M Deneke. Constructionist Learning in Software Engineering Projects. In *International Software Engineering Education Symposium*, Poznan, Poland, 1998. Scientific Publishers.

[Tho02] W. R. Thomas. *An analysis of student collaboration and task completion through project-based learning in a web-supported undergraduate course.* Dissertation, Louisiana State University, 2002.

[Tuc65] B.W. Tuckman. Developmental sequence in small groups. *Psychological Bulletin*, 63(6):384–399, 1965.

[Woo94] D.R. Woods. *Problem Based Learning: how to gain the most from PBL.* McMaster University Bookstore, Hamilton, ON, Canada, 1994.

[WU01] Laurie Williams und Richard L. Upchurch. In support of student pair-programming. In *SIGCSE '01: Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*, Seiten 327–331. ACM Press, 2001.