Effekte von Paararbeit

Tanja Bipp
Lehrstuhl für angewandte Organisationspsychologie

Andreas Lepper und Doris Schmedding
Lehrstuhl für Software-Technologie

Universität Dortmund
44221 Dortmund
{Tanja.Bipp, Andreas.Lepper, Doris.Schmedding}@udo.edu

Abstract: Wir berichten über eine umfangreiche Studie zur Paararbeit in Software-Entwicklungspraktika an der Universität Dortmund. An unserer Studie haben 13 Software-Entwicklungsteams mit insgesamt etwa hundert Studierenden teilgenommen, von denen die Hälfte ihre Projekte in Paararbeit durchgeführt hat. Die Paararbeitsgruppen haben in der gleichen Zeit nur geringfügig weniger Code produziert als die Nicht-Paararbeitsgruppen, obwohl sie an nur halb so vielen Rechnern gearbeitet haben. Dabei war der erstellte Code besser verständlich und leichter lesbar, was die Fehlersuche und die Wartung erleichtert.

1 Einleitung

Paararbeit ist ein wesentliches Merkmal von Extreme Programming (XP), einer von Kent Beck und anderen Ende der neunziger Jahre propagierten Technik der Software-Entwicklung [Be00]. Eigene Experimente mit Extreme Programming [BS04] im Software-Praktikum haben gezeigt, dass Studierende, die über wenig Erfahrung in der Software-Entwicklung verfügen, mit der Organisation und Planung eines XP-Projekts überfordert sind. Insbesondere das Einschätzen des Aufwands und das Unterscheiden von Kernfunktionalität gegenüber von „Nice-to-have"-Funktionalität fallen ihnen schwer. Da wir aber den Eindruck gewonnen haben, dass die Studierenden durchaus von Paararbeit profitieren, haben wir beschlossen, dieses Konzept im Software-Praktikum weiterhin einzusetzen und seine Vor- und Nachteile systematisch zu untersuchen.

Das Software-Praktikum ist eine Pflichtveranstaltung im 4. Fachsemester des Informatikstudiums an der Universität Dortmund. Im Team von etwa 8 Studierenden werden in 6 Wochen in der vorlesungsfreien Zeit zwei Software-Entwicklungsprojekte durchgeführt. Die eingesetzte Programmiersprache ist Java, UML [BRJ99] wird für Analyse und Design eingesetzt. Die Entwicklerteams folgen in ihren Projekten einem auf dem Unified Process basierenden Entwicklungsprozess [KSS04].

Ein Team arbeitet in Paararbeit, wenn jeweils zwei Teammitglieder gleichzeitig an einem Computer gemeinsam eine Teilaufgabe bearbeiten [WK03]. Der englische Begriff „Pair Programming" wird hier bewusst mit Paararbeit übersetzt, um zu betonen, dass

sich die Paararbeit auf alle Tätigkeiten im Softwareentwicklungsprozess erstreckt, nicht nur auf die Programmierung. In der Paararbeit werden zwei Rollen [WK03] unterschieden: Der „Driver“ arbeitet an der Tastatur, der „Observer“ liest und denkt mit, korrigiert Fehler und behält den größeren Zusammenhang im Auge. Wichtig für die Zusammenarbeit der Rollen ist eine gut funktionierende Kommunikation. Der Observer fragt nach, wenn er etwas nicht verstanden hat, der Driver muss immer in der Lage sein, seine Arbeit zu erklären. Die PartnerInnen tauschen regelmäßig die Rollen. Außerdem werden innerhalb des Teams regelmäßig die Paare neu zusammengestellt, um die Wissensweitergabe im gesamten Team zu unterstützen.

Als Vorteil dieses Konzepts wird angesehen, dass das Wissen im gesamten Team besser verbreitet wird. Ein Paararbeitsteam ist nicht so abhängig vom Ausfall einzelner Personen, die über Expertenwissen für spezielle Aufgaben verfügen, weil immer mindestens zwei Personen an allen Teilaufgaben beteiligt waren. Die Lösungen der Paare sollten besser sein, denn die Kreativität und die Erfahrungen von zwei Personen sind höher als die von einer. Der Observer führt schon während des Aufschreibens durch den Driver ein erstes Review durch, so dass viele Fehler vermieden bzw. sehr früh entdeckt werden. Deshalb ist Paararbeit als eine Qualitätssicherungsmaßnahme anzusehen.

Viele der beschriebenen Eigenschaften von Paararbeit können sich besonders positiv in studentischen Projekten auswirken. Wir beobachten bei den PraktikumsteilnehmerInnen ein extrem heterogenes Vorwissen, so dass die gezielte Unterstützung des Lernens voneinander viele Vorteile bringen könnte. Den positiven Eigenschaften von Paararbeit, die die höhere Qualität des Produkts, die bessere Integration der Teammitglieder und ihre Zufriedenheit bei der Arbeit betreffen, steht als Argument gegen Paararbeit ihre geringere Effizienz gegenüber. Wenn jeweils zwei Personen gemeinsam an einer Aufgabe arbeiten, sollte das gesamte Team etwa doppelt so viel Zeit wie ein Team aus einzeln arbeitenden Entwicklern benötigen. In Kap. 2 werden einige Ergebnisse von Studien zur Paararbeit näher betrachtet.

Um die Vor- und Nachteile von Paararbeit genauer zu untersuchen, wurden im Sommer 2004 und im Frühjahr 2005 umfangreiche Studien mit insgesamt 95 Studierenden im Software-Praktikum durchgeführt. In Kapitel 3 werden diese Studien näher beschrieben. Kapitel 4 enthält Ergebnisse zur Akzeptanz des Paararbeitskonzepts. In Kapitel 5 werden einige weitere Ergebnisse vorgestellt. Wir schließen mit einem Fazit.

2 Untersuchungen zur Paararbeit

Ein grundlegendes Experiment zur Paararbeit führte John Nosek [No98] mit 15 Vollzeit-Systemprogrammierer durch, von denen 10 in Paaren arbeiteten, während die Kontrollgruppe aus 5 einzeln arbeitenden Entwicklern bestand. Die Lösungen der Paare waren signifikant besser lesbar und besaßen eine höhere Funktionalität als die Lösungen der Nicht-Paararbeiter. Die Paare hatten mehr Vertrauen in ihre Lösungen und empfanden mehr Freude bei der Arbeit. Obwohl die Paare im Durchschnitt nur 30,2 min. für ihre Lösung benötigten, während die Nicht-Paararbeiter im Durchschnitt 42,6 min. benötigten, war der Unterschied aufgrund der kleinen Stichprobe statistisch nicht signifikant.

Vorerfahrung und Qualität der Lösung (Lesbarkeit und Funktionalität) erwiesen sich als stark korreliert, sowohl bei den Paaren (87,2%) als auch bei der Kontrollgruppe (73,5%).

An der Universität Utah wurde von Laurie Williams u. a. [Wi00] 1999 ein Software-Technik-Kurs für Fortgeschrittene mit 41 Teilnehmern durchgeführt, die in zwei Gruppen mit etwa gleichen Vorkenntnissen eingeteilt wurden. 28 Studierende bildeten die Gruppe der Paararbeiter, während 13 Studierende vier Software-Entwicklungsaufgaben alleine lösen mussten. Während Nosek [No98] nur die Programmierung in seine Studie einbezog, wurde in Utah der gesamte Entwicklungszyklus mit Analyse, Entwurf, Implementierung und Test untersucht. Entwicklungszeit, Produktivität und Qualität des Ergebnisses wurden zwischen beiden Gruppen verglichen. Die Paararbeiter erledigen ihre Aufgaben in 40-50% der Zeit der Einzelarbeiter. Die Qualität der Produkte wird daran gemessen, wie viele Testfälle durchlaufen. Bei den Paaren sind es im Durchschnitt bei jeder der vier Aufgaben mehr Tests als bei den Einzelarbeitern. Da die Programme der Paare bei gleicher Funktionalität einen geringeren Umfang besitzen, wird daraus geschlossen, dass sie eine höhere Code-Qualität besitzen. Die Paararbeiter geben an, dass ihnen die Arbeit mehr Spaß gemacht hat und dass sie mehr Vertrauen in die erarbeitete Lösung haben. Es wird zwar behauptet, dass es sich hier um eine quantitative Untersuchung handelt, aber als Ergebnisse werden nur Mittelwerte angegeben, deren statistische Signifikanz nicht überprüft wird.

McDowell u. a. stellen bei quantitativen Untersuchungen [Mc02] [Mc03] in Programmierkursen für Anfänger an der Universität von Kalifornien mit fast 600 Studierenden fest, dass insbesondere schwächere Studierende von Paararbeit profitieren. Mehr Paararbeiter als Studenten aus der Kontrollgruppe, die die Aufgaben einzeln lösen mussten, nahmen erfolgreich am Kurs teil. Auch die Langzeiteffekte waren beachtlich, mehr Paararbeiter besuchten weiterführende Informatikkurse. Hier ist zwar die Stichprobe erfreulich groß und das methodische Vorgehen korrekt, aber der Begriff der Paararbeit wird sehr eng gefasst, betrachtet wird nur das Programmieren, nicht die anderen Tätigkeiten in Software-Entwicklungsprojekten. Außerdem wurden die PartnerInnen nicht getauscht.

Paararbeit wird in dieser Studie ebenso wie bei Müller und Padberg [MP04] im Gegensatz zur Einzelarbeit gesehen, es werden keine Teams aus Paaren gebildet. In Ihrer Untersuchung an der Universität Karlsruhe haben 38 Studierende zwei kleine Programme geschrieben, für die sie durchschnittlich etwa 4 Stunden gebraucht haben. Die Hälfte der Studierenden hat in Paaren, die andere Hälfte allein gearbeitet. Die Aufgaben wurden so gestellt, dass die Funktion der Programme leicht mit automatisch generierten Testfällen überprüft werden konnte. Je mehr Testfälle bestanden wurden, desto höher wird die Qualität der Programme eingeschätzt. Müller und Padberg kommen zu dem Schluss, dass die Paare etwa 29 % mehr sinnvollen Code als die Einzelarbeiter produziert haben. Berücksichtigt man dazu noch die Codequalität, sind Einzelarbeiter nur etwa 7 % kostengünstiger als Paare.

Alle betrachteten Studien, bis auf die von Williams [Wi00], beschränken sich auf Programmierung als Entwicklungstätigkeit. Wir dagegen möchten den gesamten Entwicklungsprozess in die Untersuchung einbeziehen. Allen beschriebenen Studien ist gemeinsam, dass sie die Aufgaben jeweils entweder von einem Entwicklerpaar oder einer Ein-

zelperson erledigen lassen. Es werden keine Teams aus Einzelarbeitern mit Teams von Paaren verglichen, die die Partner tauschen. Es werden relativ kleine Aufgaben bearbeitet, deren Anforderungen klar beschrieben sind. Wir dagegen definieren eine komplexe Aufgabe, die von einem größeren Team erledigt wird, das entweder nach dem Paararbeitskonzept arbeitet oder das aus Einzelarbeitern besteht. Dieser Versuchsaufbau scheint uns sehr viel realitätsnäher zu sein als der in den oben beschriebenen Experimenten.

3 Beschreibung unserer Experimente

In unseren Experimenten [Le05] untersuchen wir die Frage, welchen Einfluss die Paararbeit auf Leistung eines Software-Entwicklungsteam hat. Als Einflussfaktoren auf die Leistung werden außer der Organisation der Arbeit als Team aus Paaren oder Einzelarbeitern noch die Vorkenntnisse der Teammitglieder, die eingesetzte Arbeitszeit und die Persönlichkeit der Mitglieder betrachtet. Die Leistung der Teams wird gemessen, indem der Umfang der erstellten Programme und ihre Qualität ermittelt werden.

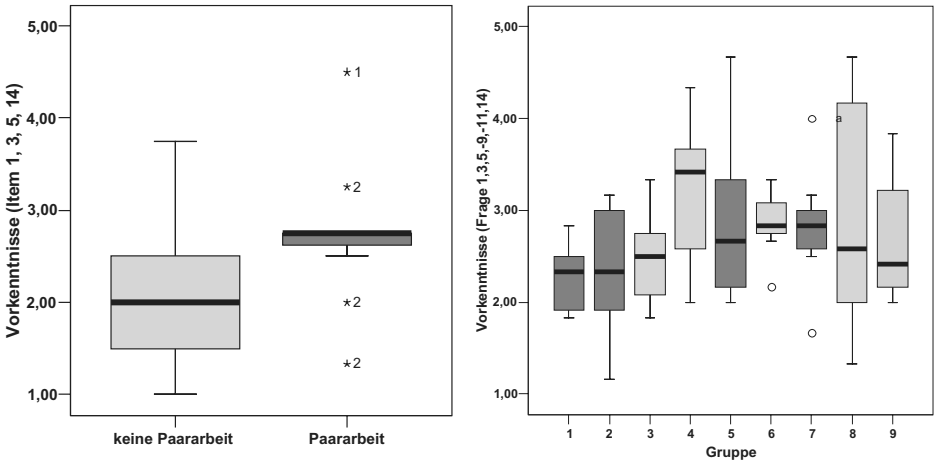

Abbildung 1:Vorkenntnisse im ersten Teil der Studie für alle Teilnehmer (links) und im zweiten Teil der Studie in den jeweiligen Gruppen (rechts)

Vorkenntnisse. An der Studie nahmen insgesamt 95 Informatik-Studierende aus dem 4. bis 6. Semester teil, die bereits erfolgreich die Vordiplomsprüfungen im Bereich Software-Entwicklung abgelegt hatten. Dennoch war das Vorwissen aufgrund von außerhalb der Universität erworbenen Kenntnissen sehr heterogen. Der Eindruck der Praktikumsveranstalter wird auch durch die eigene Einschätzung der TeilnehmerInnen bestätigt, die mit Hilfe von Fragebögen erfasst wurde (vgl. Abb. 1). Zu den Vorkenntnissen wurden eine Reihe von Fragen gestellt, die sich zu einer Skala zusammenfassen ließen (6 Fragen; Cronbachs Alpha 0,82; Beispielfrage: *„Ich habe im bisherigen Studium schon viel programmiert“*). Deshalb sollte durch eine zufällige Verteilung der Teilnehmerinnen auf die Entwicklungsteams eine gleichmäßige Verteilung von erfahrenen und weniger erfahrenen Studierenden erreicht werden.

Unsere Untersuchung besteht aus zwei Teilen. Am ersten Experiment nahmen 2004 25 Studierende teil, die nach dem Zufallsprinzip in 4 Gruppen aufgeteilt wurden. Zwei Teams arbeiteten in Paararbeit, zwei nicht. Im InWiDA-Labor [BH] der Universität Dortmund konnten wir die Gruppen bei ihrer Arbeit beobachten und Videoaufnahmen machen, sowohl von den Paararbeits- als auch von den Nicht-Paararbeitsgruppen. Aufgrund dieser Beobachtungen und der Befragung der VersuchsteilnehmerInnen haben sich viele neue Fragen ergeben, so dass Anfang 2005 ein weiteres Experiment mit 70 Teilnehmerinnen durchgeführt wurde.

Im zweiten Teil der Untersuchung wurden die 70 PraktikumsteilnehmerInnen zufällig in 9 gleich große Gruppen aufgeteilt. Die statistische Untersuchung der in den Fragebögen gemachten Angaben zu den Vorkenntnissen zeigt, dass in einzelnen Gruppen die Streuung unterschiedlich groß ist (vgl. Abb. 1 rechts), dass sich aber die Vorkenntnisse der Studierenden in den Paararbeitsgruppen (PA) nicht signifikant von den Nicht-Paararbeitsgruppen (NPA) unterscheiden (Skala von 1 (= geringe) bis 5 (= hohe Vorkenntnisse), M_{PA}=2,58; M_{NPA}=2,87; t-Test: t(60)=-1,56 p>0,05). Der Einfluss von unterschiedlichem Vorwissen auf die unterschiedlichen Leistungen der Gruppen kann also ausgeschlossen werden.

Arbeitszeiten. Fünf der 9 Gruppen wurden aufgefordert, in Paararbeit zu arbeiten, was durch die Tatsache forciert wurde, dass je zwei Entwicklern dieser Teams nur ein Arbeitsplatz zur Verfügung gestellt wurde. Die anderen Gruppen bekamen entsprechend ihrer Mitgliederzahl Arbeitsplätze zugeteilt und sie durften sich organisieren, wie sie es für sinnvoll hielten. Die Arbeitszeiten in unserem Praktikum sind nicht fest vorgegeben, sondern jedes Team arbeitet so viele Stunden, wie zur Erledigung der Aufgaben nötig ist. Für die Erfassung der Arbeitszeiten der TeilnehmerInnen wurden Stundenzettel eingesetzt. Die durchschnittliche Wochenarbeitszeit der Gruppen schwankt zwischen 22,7 und 28,8 Std. Während die Paararbeiter im Durchschnitt 26,0 Std. pro Woche gearbeitet haben, waren es bei den Nicht-Paararbeitern im Durchschnitt 25,7 Std. Der Unterschied in der Arbeitszeit der beiden Stichproben ist nicht signifikant, so dass wir im Weiteren davon ausgehen können, dass beide Gruppen etwa gleich lange gearbeitet haben.

Einhaltung des Paararbeitskonzepts. Im ersten, kleineren Teil des Experiment wurden die Gruppen während der Arbeit im Labor sehr oft beobachtet, so dass wir sicher sein konnten, dass die Paararbeiter tatsächlich zu zweit und die anderen jeweils allein an ihren Rechnern gearbeitet haben. Neben den Erledigungen von Teilaufgaben allein bzw. im Zweierteam gab es natürlich für beide Arten von Teams Sitzungen im Plenum. Die Unterstützung durch Nachbarn war weder bei den Einzelarbeitern noch bei den Paaren untersagt und konnte bei beiden Arbeitsformen beobachtet werden. Im zweiten Teil des Experiments war die Teilnehmerinnengruppe deutlich größer und eine ständige Beobachtung unmöglich. In Fragebögen wurde nachgefragt, ob zu zweit gearbeitet wurde. Vier unterschiedlich formulierte Fragen zu dieser Thematik ließen sich zu einer Skala zusammenfassen (Cronbachs Alpha 0,90; Beispielfrage: *„Wir haben in unserer Gruppe überwiegend zu zweit an einem Rechner gearbeitet."*). In einem Antwortschema von 1 (trifft überhaupt nicht zu) bis 5 (trifft völlig zu) erreichen die Paararbeiter einen Durchschnitt von 4,14, während die Nichtpaararbeiter 2,58 erreichen (siehe Abb. 2 links). Scheinbar haben auch Nicht-Paararbeiter gelegentlich zu zweit gearbeitet.

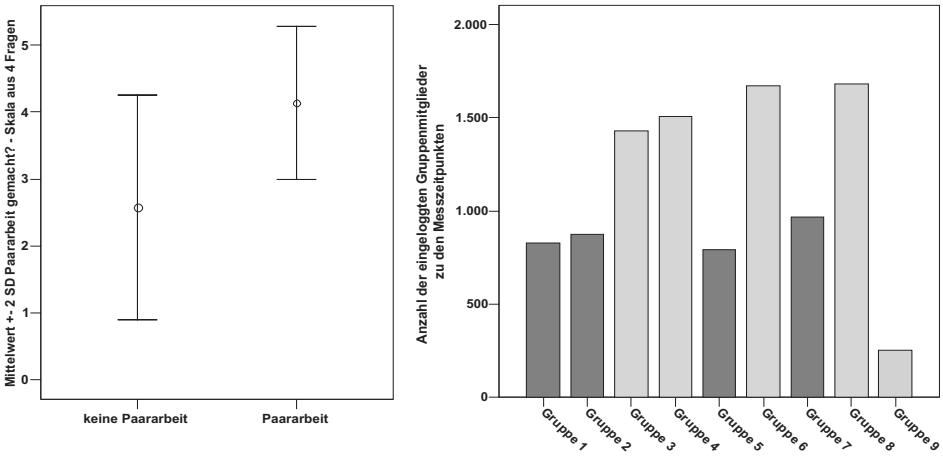

Abbildung 2 : Kontrolle der Einhaltung des Paararbeitskonzepts

Neben der Befragung der Teilnehmerinnen nach ihrem Arbeitsverhalten wurde mit Hilfe eines Unix-Skripts die Anzahl von Personen der Gruppen ermittelt, die zu den Messzeitpunkten eingeloggt waren. In Abbildung 2 (rechts) sieht man deutlich die Unterschiede zwischen den Paararbeitsgruppen (dunkel) und den Nichtpaararbeitsgruppen (hell). Die Gruppe 9 hat sich diesem Teil der Untersuchung entzogen, indem die Entwicklungstätigkeiten überwiegend auf eigenen Laptops durchgeführt wurden. Deshalb wurde diese Gruppe bei der Auswertung nicht weiter berücksichtigt.

Motivation und Persönlichkeit. Als weiterer Einflussfaktoren wurden die Motivation und die Persönlichkeit der TeilnehmerInnen auf die Leistung der Gruppen in die Untersuchung mit einbezogen. Nur mit hoher Bereitschaft zur Mitarbeit und großer individueller Leistungsbereitschaft kann ein Team eine gute Leistung erbringen. Zur Motivation wurden in einem Fragebogen 8 Fragen gestellt, die sich leider im ersten Teil der Studie nicht zu einer Skala zusammenfassen ließen. Aber aus den Beobachtungen und aus Gesprächen mit den überschaubar wenigen TeilnehmerInnen wissen wir, dass alle hoch motiviert waren. Im zweiten Teil der Studie ließen sich die Ergebnisse der 8 Fragen zusammenfassen (Cronbachs Alpha 0,82; Beispielfrage: *„Ich freue mich auf die Gelegenheit, im SoPra bisher Erlerntes anzuwenden.“*). Die Mittelwerte der einzelnen Gruppen liegen dicht beieinander zwischen 1,65 und 2,25. Ein niedriger Wert auf der Skala von 1 bis 5 zeigt eine große Bereitschaft, sich im Software-Praktikum zu engagieren. Der Test der beiden Stichproben (Paararbeiter, Nichtpaararbeiter) führte zu keinem signifikanten Unterschied. Beide Gruppen sind etwa gleich gut motiviert.

Fünf Persönlichkeitsmerkmale der TeilnehmerInnen wurden mit Hilfe des NEO-FFI-Fragebogens [BO93] erfasst. Im ersten Teil der Studie waren die Mitglieder der Paararbeitsgruppen signifikant gewissenhafter. Im zweiten Teil gab es keine signifikanten Unterschiede bezüglich der Persönlichkeitsmerkmale zwischen beiden Stichproben.

Zusammenfassung. Als Einflussfaktoren auf die Leistung eines Software-Entwicklungsteams werden Persönlichkeit und Motivation der Teammitglieder, die eingesetzte Arbeitszeit und Vorkenntnisse der Mitglieder angesehen. Soweit wir in der Lage sind, diese Faktoren zu messen, können wir im zweiten Teil der Studie keine signifikanten Unterschiede zwischen den Paararbeits- und Nichtpaararbeitgruppen feststellen. Im ersten Teil der Studie gab es bei den drei Einflussfaktoren Vorkenntnisse, Persönlichkeitsmerkmale und Arbeitszeiten so große Unterschiede, dass sie als Ursachen für unterschiedliche Ergebnisse nicht ausgeschlossen werden können.

4 Akzeptanz des Paararbeitskonzepts

Wir müssen eingestehen, dass eine der fünf Gruppen, die wir zufällig für das Paararbeitskonzept ausgewählt hatten, nicht bereit war, an dem Experiment mitzuwirken. Zwei Mitglieder in dieser Gruppe äußerten bereits im ersten Fragebogen eine extrem ablehnende Haltung gegenüber Paararbeit. Auch wenn die restlichen Mitglieder Paararbeit durchaus aufgeschlossen gegenüber standen, ließ sich das Paararbeitsexperiment in dieser Gruppe nicht durchführen. Paararbeit kann in einem Team nur funktionieren, wenn alle Mitglieder dazu bereit sind.

Die übrigen PraktikumsteilnehmerInnen in den Paararbeitsgruppen wurden zu ihren Erfahrungen mit Paararbeit in den Fragebögen einige Fragen gestellt. Die Rolle des Partners wird folgendermaßen beurteilt: Fast alle TeilnehmerInnen bewerten den Partner sehr positiv, 90% fühlen sich inhaltlich unterstützt und 50% motiviert. 38% geben an, dass der Partner sie antreibt. Sich angetrieben fühlen kann sich positiv auf die Leistung auswirken, wenn es hilft, sich besser zu konzentrieren, es kann sich aber auch negativ auswirken, wenn man sich gehetzt und kontrolliert fühlt. 5% fühlen sich behindert, ebenso viele verunsichert, äußern also eindeutig negative Gefühle.

Der Aussage „*Paararbeit hilft frühzeitig Fehler zu finden*.“ wird überwiegend zugestimmt. Die Aussage „*Ich finde es unangenehm, beim Programmieren vom Observer beobachtet zu werden.*“ trifft auf wenig Zustimmung. Ebenso wie die Aussage, dass der Partner / die Partnerin bei der Arbeit aufhält. Auf die Frage „*Bei welchen Tätigkeiten halten Sie Paararbeit für sinnvoll*?“ erhielten wir folgende Antworten: 83% empfehlen Paararbeit bei der Fehlersuche, 79% beim Programmdesign, 73% beim GUI-Design und etwa zwei Drittel beim Erstellen von UML-Diagrammen und bei komplexen Programmierarbeiten. Mehrfachnennungen waren möglich. Der größte Teil der neun aufgelisteten Tätigkeiten im Entwicklungsprozess wird für die Paararbeit empfohlen. Die geringste Zustimmung mit 36% finden „alle Programmierarbeiten“. In einem umfangreichen Projekt gibt es auch Teile, die sehr leicht zu programmieren sind, bei denen die Doppelbesetzung offenbar zumeist als überflüssig angesehen wird.

Zusammenfassend lässt sich feststellen, dass die PraktikumsteilnehmerInnen, die nach dem Paararbeitskonzept gearbeitet haben, es ganz überwiegend positiv bewerten, dass es aber eine Gruppe gab, in der das Paararbeitskonzept nicht umgesetzt werden konnte, weil zwei Teilnehmer es von vornherein ablehnten.

5 Weitere Ergebnisse

Zur Messung der Leistungen der Gruppen werden der Umfang der Projekte und ihre Qualität betrachtet.

Zum Abschluss eines Projekts im Praktikum findet eine Begutachtung der erstellten Programme durch die Praktikumsveranstalter und die Praktikumsteilnehmer selbst statt. Alle Programme waren funktionstüchtig und erfüllten die im Aufgabentext beschriebenen Mindestanforderungen. Darüber hinaus waren die Programme recht unterschiedlich gestaltet und ausgestattet und boten den Benutzern unterschiedliche Funktionalität an. In beiden Studien war jeweils eins der beiden Programme ein Spielprogramm. Teilweise besaßen sie besonders ausgefeilte Spielstrategien, eins zeigte immer alle möglichen Züge an, bei einem wurden die Spielregeln auf Wunsch angezeigt usw. Objektiv messen ließen sich die Unterschiede im Funktionsumfang nicht. Nach dem subjektiven Eindruck der Veranstalter besaßen im zweiten Teil der Studie die Programme der Nicht-Paararbeiter aber einen leicht höheren Funktionsumfang als die der Paararbeiter.

Zur Messung des Umfangs des Programmcodes wurde die Metrik „Lines of Code“ eingesetzt. In Experimenten wurde eine positive Korrelation zwischen LOC und der aufgewandten Arbeitsleistung beim Erstellen des Programms festgestellt [ED96]. Da die Klassen der grafischen Benutzungsschnittstelle in unserem Experiment mit Hilfe eines GUI-Builders erzeugt werden, bleiben diese Klassen bei unserer Umfangsmessung unberücksichtigt. Kommentarzeilen und Leerzeilen werden nicht mitgezählt.

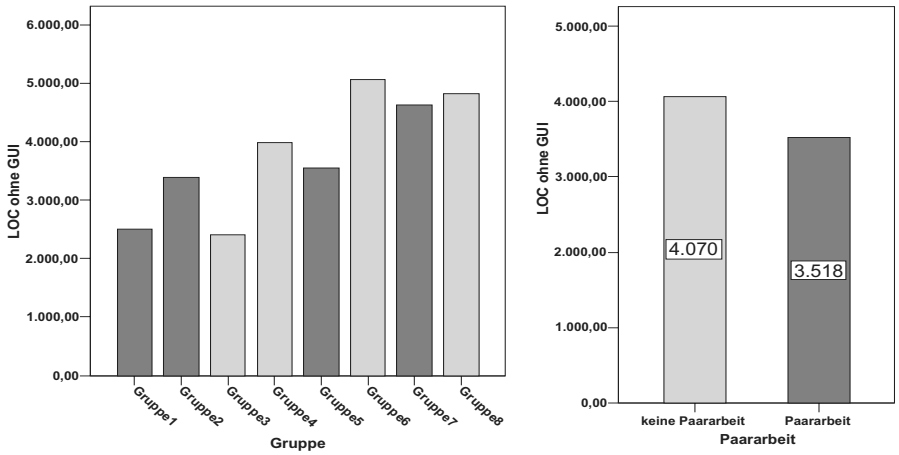

Abbildung 3: Umfang der Projekte der einzelnen Gruppen (links) bzw. durchschnittliche Projektgröße von Paararbeitern und Nichtpaararbeitern (rechts)

Abb. 3 zeigt die Ergebnisse im zweiten Teil der Studie. Der Umfang der Projekte schwankt stark zw. 2403 Zeilen (Gruppe 3) und 5068 Zeilen (Gruppe 6). Zum Cluster der größten Projekte mit etwa 5000 Codezeilen gehören eine Paararbeitsgruppe und zwei Nicht-Paararbeitsgruppen. In das Cluster der mittelgroßen Projekte mit etwa 3500 Codezeilen gehören zwei Paararbeits- und eine Nicht-Paarabeitsgruppe. Je eine Gruppe beider

Organisationsformen hat ein besonders kleines Projekt realisiert. Die Paararbeitsgruppen haben bei gleicher Arbeitszeit an halb so vielen Rechnern etwa 13,6% weniger Code produziert als die Nichtpaararbeitgruppen. Im ersten Teil der Studie haben die beiden Paararbeitsgruppen sogar etwas umfangreichere Projekte als die beiden Nicht-Paararbeitsgruppen entwickelt, was aber evtl. auf die höheren Vorkenntnisse und die höhere Gewissenhaftigkeit der Mitglieder dieser Gruppen zurückzuführen ist.

Zur Beurteilung der Qualität von Programmen werden in den anderen Studien in erster Linie Funktionstests eingesetzt (siehe Kap. 2). Das ist bei dort beschriebenen kleinen Projekten und exakt spezifizierten Anforderungen einfach möglich, gibt aber nur einen Eindruck vom Teilaspekt der funktionalen Korrektheit der erstellten Programme. Beispielsweise die Benutzungsfreundlichkeit der Produkte und die Qualität des Codes bleiben dabei außen vor.

Da wir von dem Paararbeitskonzept insbesondere Effekte auf die Qualität des erstellten Codes erwarteten, galt unser größtes Interesse diesem Qualitätsmerkmal. Insbesondere interessierte uns die Komplexität des erstellten Codes. Übermäßig komplexer Code ist schwer verständlich und schwer wartbar. Weiterhin ist zu erwarten, dass derartiger Code noch Fehler enthält, die sich erst nach längerer Laufzeit bemerkbar machen. Guter objektorientierter Code ist einfach strukturiert, da Vererbung und Polymorphie Konzepte sind, mit denen sich komplexe syntaktische Strukturen vermeiden lassen.

Aufbauend auf Arbeiten von Chidamber und Kemerer [CK94] bieten heute viele Software-Entwicklungswerkzeuge die Möglichkeit spezielle Metriken zur Komplexitätsmessung von objektorientierten Programmen einzusetzen. Dabei werden die Vererbungshierarchie, die Kopplung der Objekte und die Kohäsion betrachtet. Beim Einsatz von Vererbung, gemessen mit den Metriken DIT (Depth of Inheritence Tree) und NOC (Number of Children), unterscheiden sich die Paararbeitsgruppen nicht von den anderen. Die Metrik CBO (Coupling between Objects) macht zwar Unterschiede zwischen den einzelnen Gruppen deutlich, es gibt aber keinen eindeutigen Trend für die Paararbeiter.

Unterschiede werden aber bei den „echten“ Komplexitätsmaßen, LCOM (Lack of Cohesion in Methods), RFC (Response for a Class) und WMC (Weighted Methods per Class) deutlich (siehe Abb. 4). Ein hoher LCOM-Wert ist ein Zeichen dafür, dass Programmteile zu einer Klasse zusammengefasst wurden, die nicht zusammen gehören. Ein hoher RFC-Wert weist auf eine starke Kopplung der betrachteten Klasse mit anderen hin. Das wirkt sich negativ auf die Testbarkeit und Änderbarkeit der Klasse aus. Ein hoher WMC-Wert einer Klasse, der die Anzahl der möglichen Pfade im Kontrollflussgraphen angibt, deutet darauf hin, dass eine Klasse nur schwer zu testen ist. Die drei Nicht-Paararbeitsgruppen 4, 6 und 8 liefern hier Spitzenwerte, während die Paararbeiter geringe Werte erzielen. Die Nichtpaararbeitsgruppe 3, die eines der Programme mit dem geringsten Umfang geschrieben hat, erzielt ebenfalls geringe Komplexitätswerte.

Obwohl die Unterschiede zwischen den Paararbeitsgruppen gegenüber den anderen Gruppen durchaus deutlich sind, sind sie statistisch nicht signifikant, da die Stichprobe zu klein ist, um allgemeingültige Aussagen zu treffen. Wir können hier also nur einen Trend zu weniger komplexen Programmen beim Einsatz von Paararbeit sprechen.

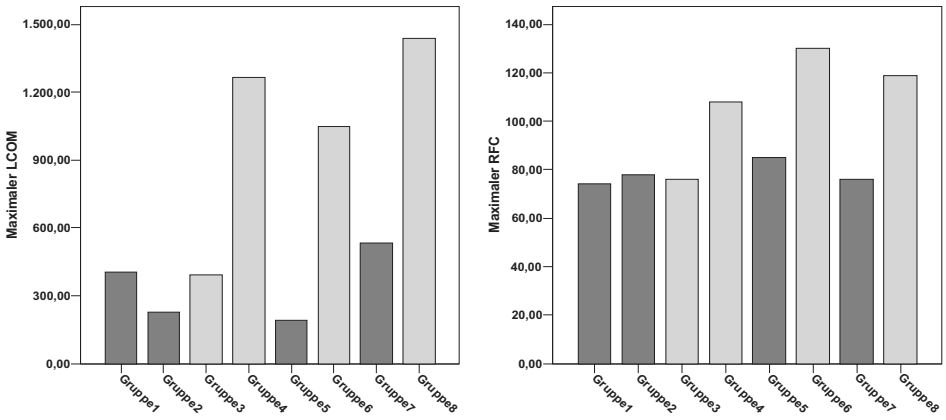

Abbildung 4: Die Komplexitätsmaße LCOM (links) und RFC (rechts)

Außer durch Metriken wurde im zweiten Teil der Studie der Programmcode noch durch Informatikfachleute, Mitarbeiter des Lehrstuhls für Software-Technologie des Fachbereichs Informatik, beurteilt. Angelehnt an Kriterien für „übel riechenden“ Code, wie er von Fowler [Fo00] beschrieben ist, wurde ein Kriterienkatalog aufgestellt, der die Fachleute bei ihrer Beurteilung unterstützte. Besonders berücksichtigt werden sollte die Verständlichkeit und Lesbarkeit, die zum Beispiel durch aussagekräftige Bezeichner und Kommentare erreicht wird, deren Qualität sich durch Metriken nicht messen lässt. Unsere Experten bekamen jeweils zwei Klassen von mehreren Gruppen vorgelegt, von denen sie nicht wussten, ob sie nach dem Paararbeitskonzept arbeiteten oder nicht. Der Code der Paararbeitgruppen wurde als geringfügig besser lesbar eingeschätzt.

Da in den Paararbeitsgruppen nicht nur die Programmierung sondern alle Aufgaben der Software-Entwicklung von Paare erledigt wurden, wäre es interessant, auch die in den frühen Phasen erstellten Diagramme und Dokumente in den Vergleich mit einzubeziehen. Da die Beurteilung der Qualität von UML-Diagrammen und Dokumenten noch schwieriger als die Beurteilung von Software-Qualität ist, wurde hier darauf verzichtet. Das ist aber Thema einer aktuellen Untersuchung.

Wir konnten im Labor beobachten, dass in den Nicht-Paararbeitsgruppen die Gefahr besteht, schwächere Studierende aus der Gruppe auszuschließen, so dass sie praktisch keinen Beitrag zum Projektfortschritt leisten. In diesen Gruppen arbeiteten die jeweils leistungsstärksten Studierenden an den wichtigsten Teilaufgaben des Projekts, während die weniger erfahrenen Studierenden weniger wichtige Zuarbeiten leisteten, einfache Klassen implementierten, Handbuch schrieben, Test durchführten usw. In den Paararbeitsgruppen dagegen wurden alle Teilaufgaben von Paaren erledigt, die sich aus unterschiedlich leistungsstarken Studierenden zusammengesetzt haben.

Neben der Wirkung des Paararbeitskonzepts auf die Qualität der erstellten Software interessierte uns auch ihre Auswirkung auf die Arbeitsverteilung unter den Teammitgliedern und an den einzelnen Teilen des Projekts. Zur Untersuchung dieser Aspekte wurden

Fragebogeninstrumente eingesetzt. Bei der Beurteilung der Aussage „*In unserer Gruppe war die Arbeitsbelastung ungleich*." stimmen die Paararbeiter auf einer fünfstufigen Skala signifikant weniger zu als die anderen (M_{PA}=2,68; M_{NPA}=3,23; t-Test: t(59)=-2,19 p=0,03). Die Paararbeiter haben es also besser verstanden, die Arbeitsbelastung gleichmäßig zu verteilen.

Auch die Aussage „*Teile unseres Projekts kenne ich kaum*." fand bei den Nichtpaararbeitern (NPA) signifikant mehr Zustimmung als bei den Paararbeitern (PA) (5-Stufige Skala, M_{PA}=2,26; M_{NPA}=3,17; t-Test: t(59)=-3,35 p=0,01). Das Wissen über das Gesamtprojekt konnte also durch Paararbeit besser im gesamten Team verbreitet werden. Unterstützung fand dieser Befund in den Daten der ersten Studie, in denen unabhängig von der vorliegenden Programmiererfahrung, die Teilnehmer in den Paararbeitsgruppen von einer höheren Klarheit in Bezug auf die zeitliche Strukturierung der Arbeitsabläufe berichteten, als die Teilnehmer in den Nicht-Paararbeitsgruppen (eingesetzte Skala: Klarheit über zeitliche Arbeitsabläufe [SH98], 3 Fragen, Cronbachs Alpha 0,73; Beispielfrage: „*Ich wusste genau, in welcher zeitlichen Abfolge ich meine Arbeit zu erledigen hatte.*", Antwortformat 1 (stimme überhaupt nicht) bis 7 (stimme vollständig zu)). Den Teilnehmern in den Paararbeitsgruppen (M_{PA}=5,64) war demnach klarer, in welcher Reihenfolge Arbeiten innerhalb des Projektes zu erledigen waren oder wann welche Arbeitshandlungen durchzuführen waren, als den Teilnehmern in den anderen Gruppen (M_{NPA}=5,15; F(1, 22)=6,19, p<0,05).

6 Fazit

Es stellt sich die Frage, warum 8 Personen an 4 Arbeitsplätzen nicht doppelt so viel Zeit für ein Projekt benötigen wie 8 Personen an 8 Arbeitsplätzen. Wir schließen aus unseren Befragungen und Beobachtungen, dass Paararbeiter sehr davon profitieren, dass sie bessere Kenntnis von allen Teilen des Projekts besitzen und ihre Zeit intensiver nutzen, weil sie konzentrierter arbeiten und sich nicht so leicht ablenken lassen. Bei Problemen erfahren sie Unterstützung durch ihren Partner. Bei der Entwicklung komplexer Software wird sehr viel Zeit mit der Suche nach Fehlern aufgewendet. Gerade hierbei können Paararbeiter zu zweit schneller sein. Die Zusammenarbeit mit dem Partner wird von fast allen Paararbeitern sehr positiv bewertet. Vorteile ziehen Paararbeiter auch aus der Qualitätsverbesserung ihres Codes, der weniger komplex, also einfacher strukturiert und besser lesbar ist. Das macht die Fehlersuche einfacher und bringt einen Zeitgewinn.

Die überwiegende Mehrheit der Paararbeiter stellt in den Befragungen den Nutzen von Paararbeit heraus, während nur wenige sich negativ äußern. In der Gruppe, die sich vom Experiment ausgeschlossen hat, waren von Anfang an zwei Personen so stark gegen Paararbeit eingestellt, dass die Umsetzung keinen Sinn gemacht hätte.

Die für eine Lehrveranstaltung äußerst spannende Frage, ob Paararbeiter durch die intensive Zusammenarbeit mit einem Partner zu einem höheren Lernerfolg führt, lässt sich durch Selbsteinschätzung mit Hilfe von Fragebögen leider nicht klären. Alle TeilnehmerInnen (Paararbeiter und Nicht-Paararbeiter) behaupten viel bzw. sehr viel gelernt zu haben. Hier zeigten sich keine signifikanten Unterschiede zwischen beiden Stichproben.

Da aber die Paararbeiter angeben, höhere Kenntnisse über das Projekt erworben zu haben, kann man davon ausgehen, dass sie mehr über Software-Entwicklung gelernt haben.

Wenn die Produktivitätsverluste durch Paararbeit nur gering sind und ihnen auf der anderen Seite aber Qualitätsverbesserungen beim Code und eine höhere Integration alle Teammitglieder, auch der weniger erfahrenen, in das Projektgeschehen gegenüberstehen, lohnt sich Paararbeit aus unserer Sicht in einer Lehrveranstaltung auf jeden Fall.

Die Durchführung der Untersuchung im Rahmen der universitären Ausbildung bot die Möglichkeiten, unter weitgehend kontrollierten Bedingungen mit relativ vielen Versuchsteilnehmern in mehreren Entwicklungsteams zu arbeiten, die gleichzeitig die gleichen Projekte durchgeführt haben. Wie weit die Ergebnisse auf außeruniversitären Entwicklerteams übertragbar sind, bedarf weiterer Überprüfung.

Literaturverzeichnis

[Be00] Beck, K.: Extreme Programming explained: Embrace Chance. Addison Wesley, 2000.

[BS04] Beckmann, I.; Schmedding, D.: Experimente mit XP in der Lehre. GI Jahrestagung (2, 2004), S. 122-126.

[BH] Bipp, T.; Hüvelmeyer, J.: Das INWIDA Labor, http://www.inwida.uni-dortmund.de

[BRJ99] Booch, G.; Rumbaugh, J.; Jacobson I.: The Unified Modeling Language – User Guide. Addison Wesley, 1999.

[BO93] Borkenau, P.; Ostendorf, F.: NEO-Fünf-Faktoren Inventar, Handanweisung, Hogrefe Verlag für Psychologie, 1993.

[CK94] Chidamber S. R.; Kemerer C. F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, Vol. 20, No. 6, 1994; pp. 476-493.

[ED96] Ebert, C.; Dumke, R.: Software-Metriken in der Praxis. Springer, 1996.

[Fo00] Fowler, M.: Refactoring – Improving the Design of Existing Code. Addison-Wesley, 2000.

[KSS04] Kopka, C.; Schmedding, D.; Schröder, J.: Der Unified Process im Grundstudium - Didaktische Konzeption, von Lernmodulen und Erfahrungen. DeLFI 2004; pp. 127-138.

[Le05] Lepper, A.: Eine empirische Studie über Paararbeit in der Softwaretechnik. Diplomarbeit am Fachbereich Informatik der Universität Dortmund, 2005.

[No98] Nosek, J. T.: The Case for Collaborative Programming. Communications of the ACM, Vol. 41, No. 3, 1998.

[Mc02] McDowell, C.; Werner, L.; Bulock, H.; Fernald, J.: The Effects of Pair-Programming on Performance in an Introductory Programming Course. ACM SIGCSE Bulletin , Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, Vol. 34, No. 1, 2002.

[Mc03] McDowell, C.; Werner, L.; Bulock, H.; Fernald, J.: The impact of pair programming on student performance, perception and persistence. Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, 2003; pp. 602 – 607.

[MP04] Müller, M. M.; Padberg, F.: An Empirical Study about the Feelgood Factor in Pair Programming. In Intern. Symposium on Software Metrics, Chicago, Sep. 2004.

[SH98] Schmidt, K.-H.; Hollman, S.: Eine deutschsprachige Skala zur Messung verschiedener Ambiguitätsfacetten bei der Arbeit. Diagnostica, 44, 1998; S. 21-29.

[Wi00] Williams, L.; Kessler, R. R.; Cunningham, W.; Jeffries, R.: Strengthening the Case for Pair-Programming. IEEE Software, Juli/August 2000.

[WK03] Williams, L.; Kessler, R. R.: Pair Programming Illuminated. Addison Wesley, 2003.