Aspekte des Pair Programming

Ausgewählte empirische Untersuchungen

Seminar "Ausgewählte Beiträge zum Software Engineering"
WS 2004/2005

Was will dieser Vortrag?

- Einführen in Pair Programming und seine softwaretechnischen und organisatorischen Aspekte
  - Was wird durch Pair Programming beeinflusst?
  - Was beeinflusst Pair Programming?
- Zusammentragen einiger ausgewählter Ergebnisse über die Auswirkungen/Aspekte von Pair Programming
  - Dabei wird versucht, "Schubladen" für die Ergebnisse zu finden und
  - gegen Ende die äußere Gültigkeit hinterfragt.
- Finden eines Ansatzpunktes für die Beschreibung offener und interessanter Fragestellungen:
  - Herangehensweisen bei der Untersuchung des Objektes aufzeigen. Z.B. die Beachtung von
    - speziellen Situationen
    - speziellen Personentypen
  - Ist es sinnvoll, eine Entscheidungsmatrix für die Anwendung und Konstruktion von Pair Programming zu entwickeln?
    - Z.B. mit so unterschiedlichen Parametern wie Entwicklungsprozess, Phase des Entwicklungszyklus, Art des Problems, Persönlichkeit der Mitarbeiter, Skills der Mitarbeiter, Time-To-Market, etc.

Aufbau

- Begriffsklärung
- Historie
- Motivation von Fragestellungen bzgl. Pair Programming
- Übersicht der vorgestellten Artikel
- Vorstellung der Artikel
  - Zuerst eine Klassifizierung
  - Nachfolgend empirische Ergebnisse
  - Jeweils Hinterfragen der Ergebnisse
- Zusammenfassung der Ergebnisse
- "Und nun?“
  - Forschungsansätze?

Die Materie oder "Worum geht‘s?" (1)

- **Pair Programming** (PP) (nach Laurie Williams, Robert R. Kessler, Ward Cunningham und Ron Jeffries; 2000):
  - Zwei Entwickler stellen zusammen ein Artefakt (Design, Algorithmus, Code etc.) her.
  - Sie agieren wie ein zusammenhängender, intelligenter Organismus, der mit einem Verstand arbeitet und verantwortlich für jeden Aspekt des Artefakts ist.
  - Ein Teilnehmer ist der **Driver**:
    - Er hat die Kontrolle über den Stift, die Maus bzw. die Tastatur.
    - Er schreibt das Design oder den Code.
  - Der anderer Teilnehmer ist der **Observer**:
    - Er beobachtet kontinuierlich die Arbeit des Driver.
    - Er sucht nach Fehlern, denkt über Alternativen nach, beobachtet die Ressourcen, betrachtet "strategische" Auswirkungen.
  - Die Rollen werden bewusst periodisch getauscht.
  - Beide sind zu jeder Zeit **gleichberechtigte und aktive Teilnehmer** des Prozesses und teilen sich den Besitz am Arbeitsergebnis.

Die Materie oder "Worum geht‘s?" (2)

- Laurie Williams, Robert R. Kessler, Ward Cunningham und Ron Jeffries teilen Pair Programming in unterschiedliche Aktivitäten auf:
  - "pair-analysis"
  - "pair-design"
  - "pair-implementation"
  - "pair-testing"
- Pair Programming kann also als weit aus mehr als "Coding" angesehen werden.
- Allerdings wird in den meisten Artikeln das "Coding" in den Mittelpunkt gestellt!

Pair Development?

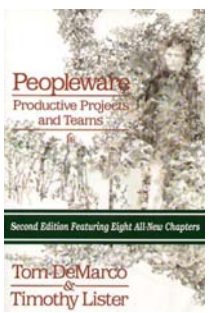
Die Materie: Historie (1)

- Historie der Begriffsbildung (1)
  - ~ 1953 Fred Brooks
    - Führte Pair Programming durch, ohne eine Namen dafür zu haben

– "Fellow graduate student Bill Wright and I first tried pair-programming when I was a grad student. **We produced 1500 lines of defect-free code; it ran correctly first try**.

I encourage my own computer architecture students to work in pairs on all design projects. **I am firmly convinced that it is not only as productive per person-hour, but also much more educational**."

  - 1987 Tom De Marco, Timothy Lister: Peopleware
    - Der Begriff *"jelled teams"* (eingeschworenes Team) wird verwendet:



– Dies ist eine Gruppe von Personen, die so enge Beziehungen untereinander pflegt, **dass das Ganze größer als die Summe seiner Teile ist**.
– Die Ergebnisse solcher Teams sind viel "größer" als die Ergebnisse der gleichen Personen, wenn jeder für sich allein arbeitet.
– **Der Spaß an der Arbeit ist bei solchen Teams größer** (als es die Natur der Arbeit anscheinend rechtfertigt).

Die Materie: Historie (2)

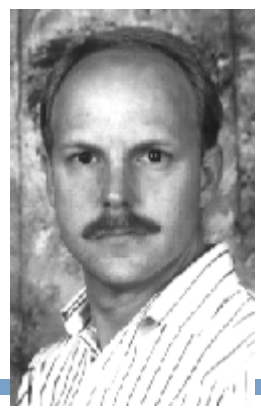
- Historie der Begriffsbildung (2)
  - 1995 Larry L. Constantine: Constantine on Peopleware

    - Der Begriff *"dynamic duo"* wird verwendet
      - Berichtet, dass die Beobachtung der Duos zeigt, dass diese schneller Code erzeugen und dieser weniger "Bugs" enthält

  - 1995 Jim Coplien: (*Bell Laboratories*)
    - Veröffentlicht das organisatorische Pattern **Developing in Pairs**
      - **Problem:**
        - » People are scared to solve problems alone.
      - **Context**:
        - » Code ownership has been identified and development is proceeding.
      - **Solution:**
        - » **Pair compatible designers** to work together; together, **they can produce more than the sum of the two individually**.

Die Materie: Historie (3)

- Historie der Begriffsbildung (3)
  - 1995/96 Extreme Programming (XP) beginnt zu entstehen (Kent Beck, Ward Cunningham und Ron Jeffries)
    - Es enthält als zentrale Praktik Pair Programming
    - 1995 WyCash-Projekt (Portfolio Management System)
      - Unter Beteiligung von Ward Cunningham
      - Alle XP-Praktiken (nur noch nicht unter dem Namen)
    - 1996 C3 Projekt (Chrysler Comprehensive Compensation)
      - Kent Beck und Ron Jeffries sind Coaches (ab 1996)
      - Gehaltsabrechnung
      - Launch 1997
  - 1999 Kent Beck: Extreme Programming Explained: Embrace Change


Kent Beck


Ward Cunningham


Ron Jeffries

Das Problem oder "Was wir wissen wollen!" (1)

- Es gibt (viele) persönliche positive Aussagen zum PP:
  - Z.B. von Kent Beck:

"Meiner Erfahrung nach ist es produktiver, paarweise zu programmieren, als die Arbeit zw. zwei Programmierern aufzuteilen und die Ergebnisse dann zusammenzuführen."

  - Z.B. von Ron Jeffries:

"I'm not entirely stupid. I noticed very quickly that this most junior of programmers was actually helping me! Me! Can you believe it? Me! That has been my experience every time thereafter, in pair-programming. Having a partner makes me a better programmer."

  - Z.B. Anonym (in Umfrage von L. Williams):

"I strongly feel pair-programming is the primary reason our team has been successful. It has given us a very high level of code quality (almost to the point of zero defects). The only code we have ever had errors in was code that wasn't pair programmed … we should really question a situation where it isn't utilized."

Das Problem oder "Was wir wissen wollen!" (2)

- Geläufige Annahmen über die "positiven" Eigenschaften von PP sind:

Kommunikation wird erhöht	Prozess-verbesserung
Die Performance wird erhöht	
Neulinge können v. Kennern effizient in den Code eingearbeitet werden	
Entwürfe werden schneller und besser durchdacht	Produkt-verbesserung
"Fehler" werden leichter erkannt	
Die Fehlerrate wird deutlich abgesenkt	
Die Algorithmen/der Code wird verbessert (knapper u. effizienter Code)	
Der Spaß an der Arbeit wird erhöht	Soziale Verbesserung
Das Vertrauen in die eigenen Programmierung wird verbessert	

Das Problem oder "Was wir wissen wollen!" (3)

- Für welche der Annahmen gibt es nun (empirische) Belege?
  - Unter welchen Bedingungen?
- Gibt es quantitative Aussagen?
- Und ist die "Paarung" in jeder Phase gleich sinnvoll?
  - An welcher Stelle eines Entwicklungszyklus fängt PP eigentlich an bzw. hört PP auf?
  - Wie interagiert PP mit dem gesamten Entwicklungsprozess?
    - Muss PP in XP eingebettet werden?
- Wie sollten Paare eigentlich gebildet werden?
  - Einfluss der Eigenschaften der Personen
  - Einfluss der Eigenschaften der Aufgabenstellung
  - Wie langlebig sollte ein solches Paar eigentlich sein?

Das Problem oder "Was wir wissen wollen!" (4)

Programmierer arbeiten ungern zusammen
mit einer anderen Person, da
dies ihre Arbeit verlangsamt!

Widerstände
gegen
Pair Programming?

Programmierer sind ein knappe Ressource!

Programmieren wird traditionell
als individuelle Tätigkeit gelehrt
und praktiziert!

Die Quellen oder "Wer liefert Anworten?" (1)

- [Nosek98] J.T. Nosek: The case of collaborative programming
  - Veröffentlicht: 1998 in *Communications of the ACM*
  - Inhalt:
    - Beschreibung und Auswertung einer Feldstudie (+ Befragung) bzgl. PP
    - Untersuchte Eigenschaften:
      - Lesbarkeit und Funktionalität des Codes
      - Problemlösungszeit
      - Vertrauen der Programmierer in ihre Lösung

Die Quellen oder "Wer liefert Anworten?" (2)

- [WilKesCun00] L. Williams, R. R. Kessler, W. Cunningham und R. Jeffries: Strengthening the Case for Pair-Programming
  - Veröffentlicht: 2000 in *IEEE Software*
  - Inhalt:
    - Zusammenfassung mehrerer "älterer" PP-Ergebnisse
    - Schwerpunkt auf einem kontrollierten Experiment mit Studierenden bzgl. PP
      - Untersuchte Eigenschaften: Produktivität (Zeit) und Qualität der Ergebnisse

Assistant Professor Dr. Laurie Williams
North Carolina State University

Professor Robert R. Kessler
University of Utah

Ward Cunningham

Ron Jeffries
Independent consultant

Die Quellen oder "Wer liefert Anworten?" (3)

- [CocWil01] Alistair Cockburn, and Laurie Williams: The Costs and Benefits of Pair Programming
  - 2001 in *eXtreme Programming and Flexible Processes in Software Engineering XP2000*
  - Inhalt:
    - Klassifizierung von Aspekten des PP
    - Begutachtung von Interviews und Experimenten bzgl. PP

Assistant Professor Dr. Laurie Williams
North Carolina State University

Dr. Alistair Cockburn
North Carolina State University

Die Quellen oder "Wer liefert Anworten?" (4)

- [KatWilWie04] N. Katira, L. Williams, E. Wiebe, C. Miller, S. Balik, and E. Gehringer: On Understanding Compatibility of Student Pair Programmers
  - Veröffentlicht: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education
    - Inhalt:
      - Beschreibung einer Befragung (Teilnehmer waren Studierende bestimmter Kurse) bzgl. PP
      - Fokus auf den möglichen Faktoren, die Einfluss auf Kompatibilität der (studentischen) Paarprogrammierer haben

Assistant Professor Dr. Laurie Williams
North Carolina State University

Klassifikation der Eigenschaften von PP nach [CocWil01] (1)

- Cockburn und Williams beschreiben in [CocWil01] acht Aspekte, die bei der Erforschung von PP bzgl. der **softwaretechnischen** und der **organisatorischen Effektivität** betrachtet werden können:
  1. Wirtschaftlichkeit (*Economics*)
     - Hypothesen:
       - Nur geringe Entwicklungskostenerhöhung für das Hinzufügen einer weiteren Person
         ⇨ [WilKesCun00], [Nosek98]
       - Der resultierende Code hat weniger Defekte ⇨ [WilKesCun00]
       - Die Ersparnisse beim Entfernen von Defekten übersteigen die Entwicklungsmehrkosten
  2. Zufriedenheit (*Satisfaction*)
     - Hypothese:
       - Personen, die in Paaren programmieren, finden dies angenehmer als alleine zu arbeiten ⇨ [WilKesCun00], [Nosek98]

Klassifikation der Eigenschaften von PP nach [CocWil01] (2)

3. Qualität des Designs (*Design quality*)
   - Hypothese:
     - Paare erzeugen kürzeren Code als Entwickler, die alleine programmieren (Indikator für bessere Design) ⇨ [WilKesCun00], [Nosek98]
4. Kontinuierliche Reviews (*Continuous Reviews*)
   - Hypothesen:
     - Das Beieinandersitzen des PP fungiert als fortwährender Design- und Code-Review und führt zu effizienter Defektbeseitigung
5. Problemlösung (*Problem solving*)
   - Hypothese:
     - Teams haben die Fähigkeit, ("unlösbare") Probleme schneller zu lösen

Klassifikation der Eigenschaften von PP nach [CocWil01] (3)

6. Lernen (*Learning*)
   - Hypothese
     - Die Mitglieder eines Paares lernen voneinander
7. Teambildung und Kommunikation (*Team Building and Communication*)
   - Hypothese:
     - Die Mitglieder lernen miteinander zu diskutieren und zu arbeiten. Dies Verbessert die Kommunikation und die Effektivität ⇒ [KatWilWie04]
8. Mitarbeiter und Projektmanagement (*Staff and Project Management*)
   - Hypothese
     - Da viele Personen mit jedem Teil des Codes vertraut sind, vermindert PP das Risiko durch Mitarbeiterverlust.

[Nosek98] J.T. Nosek: The case of collaborative programming: Hypothesen (1)

- J.T. Nosek stellt in seinem Artikel vier zu untersuchende Hypothesen bzgl. PP auf:
  - Hypothese 1: Beim Pair Programming wird eine lesbarere und bzgl. des gestellten Programmierproblems funktionalere Lösung erstellt.
  - Anmerkungen zur Hypothese 1:
    - **Lesbarkeit**:
      - *"the readability of the proposed solution, that is, the degree to which the problemsolving strategy could be determined from the subject's work"*
      - Definition einer Variable READABILITY:
        - » O bei einer völlig unlesbaren Lösung
        - » 2 bei einer komplett lesbaren Lösung
        - » 1 sonst
    - **Funktionalität**:
      - *"the functionality of the proposed solution, that is, the degree to which the strategy accomplishes the objectives stated in the problem description"*
      - Definition einer Variable FUNCTIONALITY:
        - » Summe zwischen 0 und 6, die zu der Erfüllung der (auf einer folgenden Folien beschriebenen) Ausgabeanforderungen der Aufgabe korrespondiert (0: Ziel komplett verfehlt, 6: Ziel komplett erreicht)
    - **Die Lösungen wurden später separat von zwei Personen beurteilt**.

[Nosek98] J.T. Nosek: The case of collaborative programming: Hypothesen (2)

- Hypothese 2: Gruppen benötigen im Durchschnitt weniger Zeit für die Problemlösung als alleine arbeitende Personen.
- Anmerkungen zur Hypothese 2:
  - In der Hypothese wird der verwendetet Zeitbegriff nicht konkretisiert. Handelt es sich bei der Gruppenzeit um
    - die Summe der Einzelzeiten oder
    - um das verstrichene Zeitintervall?
- Hypothese 3: Programmierer in Paaren haben direkt nach der Problemlösung ein höheres Level von Vertrauen in ihre Arbeit (CONFID) und äußern eine größerer Freude am Lösungsprozess (ENJOY).

Neuer Aspekt

- Hypothese 4: Programmierer mit einer größeren Anzahl von Jahren an Erfahrung arbeiten besser als Programmierer mit weniger Jahren an Erfahrung.

[Nosek98] J.T. Nosek: The case of collaborative programming : Aufbau und Durchführung (1)

- Versuchsaufbau und -durchführung:
  - ~Feldstudie (+ Befragung)
  - Teilnehmer: 15 erfahrene Systementwickler einer Firma, die sich mit *"program trading"* beschäftigt
  - Aufgabe: Lösen einer anspruchsvollen, für das Unternehmen wichtigen Programmieraufgabe
    - Entwicklung eines Skriptes zur Konsistenzprüfung einer Datenbank (DBCC)
      - Programm soll Ausgaben in Datei und am Bildschirm produzieren
    - DBCC werden in diesem Unternehmen normalerweise als so kritisch für den Erfolg und so weit über den Skills im Haus angesehen, dass Berater von außerhalb hierfür herangezogen werden.
    - Entwicklung unter X-Windows und C
    - Entwickler arbeiteten in ihrer eigenen Umgebung mit ihrem eigenen Equipment (bei allen eine Sun Microsystems SPARCstation)

[Nosek98] J.T. Nosek: The case of collaborative programming: Aufbau und Durchführung (2)

- Teilnehmer wurden zufällig auf Versuchs- und Kontrollgruppe verteilt
  - 5 Paare
  - 5 Einzelprogrammierer
- Bearbeitungszeit max. 45 Minuten (Zeitmessung via Stoppuhr; ausschlaggebend war die verstrichene Zeit)
- Anschließend sollte jeder Teilnehmer einen Fragenkatalog bzgl. der Punkte CONFID und ENJOY beantworten
  - Leider keine genaueren Angaben zu den Fragen vorhanden (z.B. Skalenangabe)

[Nosek98] J.T. Nosek: The case of collaborative programming: Ergebnisse (1)

- Ergebnisse (Mittelwerte, zweiseitiger t-Test)

Comparison of Individual and Team Measurements

Variable	Control Group (Individuals) mean (st. dev.)	Experimental Group (Teams) mean (st. dev.)
Performance Scores:	n = 5	n = 5
READABILITY	1.40 (0.894)	2.00 (0.000)
FUNCTIONALITY	4.20 (1.788)	5.60 (0.547)*
SCORE	5.60 (2.607)	7.60 (0.547)*
TIME (minutes)	42.60 (3.361)	30.20 (1.923)
Satisfaction Measures:	n = 5	n = 10
CONFID	3.80 (2.049)	6.50 (0.500)*
ENJOY	4.00 (1.870)	6.60 (0.418)*

*less than 1 in 20 that results are due to chance

Skala?

[Nosek98] J.T. Nosek: The case of collaborative programming: Ergebnisse (2)

- Auswertung der Ergebnisse
  - Hypothese 1 (SCORE = READABILTY + FUNCTIONALITY): Gilt hiernach als statistisch signifikant bestätigt
  - Hypothese 2 (TIME): Gilt hiernach *nicht* als statistisch signifikant bestätigt:
    - Obwohl die durchschnittliche Fertigstellungszeit bei den Einzelprogrammierern mehr als 12 Minuten höher lag (41%), gibt es eine Wahrscheinlichkeit von mehr als 0,05%, dass das Ergebnis zufällig ist.
  - Hypothese 3 (CONFID/ENJOY): Gilt hiernach als statistisch signifikant bestätigt

[Nosek98] J.T. Nosek: The case of collaborative programming: Beurteilung

- Das Experiment ist deshalb besonders wertvoll, weil
  - "echte" Programmierer eine
  - "echte" Programmieraufgabe durchführen
- Leider sagt der Artikel nichts zu folgenden Fragen:
  - Wie wurden die Paare gebildet?
    - Die Hypothese 4 lässt vermuten, dass es erfahrenere und weniger erfahrenere Programmierer gab.
  - Hatten die Teilnehmer Erfahrung mit Pair Programming?
  - Wurden die Teilnehmer auf das Pair Programming vorbereitet?
  - Wie war die Einstellung der Teilnehmer zum Pair Programming vor dem Experiment ?
  - In wie weit spielte Design beim Lösen der Aufgabe überhaupt eine Rolle?
    - Welche Phasen wurden beim PP durchlaufen?
  - Ist es während der Programmierung zu Informationsaustausch gekommen?

[WilKesCun00] Williams, Kessler, Cunningham, Jeffries: Strengthening the Case for Pair-Programming: Inhalt

- In [WilKesCun00] werden nicht explizit Hypothesen formuliert und überprüft.
- Der Inhalt wird vielmehr wie folgt beschrieben:

  "The purpose of this article is to demonstrate, through anecdotal, qualitative and quantitative evidence, that incorporating pair-programming into a software development process will help yield software products of better quality in less time with happier, more confident programmers."

- Hierzu dienen hauptsächlich zwei Teile des Artikels:
  1. Beschreibung eines **PP-Experimentes** und seiner Ergebnisse
  2. Zusammenführung diverser Aussagen die PP als **Pair Development** betrachten

[WilKesCun00] Das Experiment : Versuchsaufbau und –durchführung (1)

- Versuchsaufbau und -durchführung:
  - Kontrolliertes Experiment + Befragung (1999)
  - Teilnehmer: 41 Studenten des *"Senior Software Engineering course"* der University of Utah
    - Alle hatten die selben Kurse besucht
    - Studenten hatten signifikante Codierpraxis
    - Vorweg Teilnahme an einer Diskussion über die Pros und Contras zum PP
  - Aufgaben:
    - 4 Aufgaben über einen Zeitraum von 6 Wochen
    - Dieselbe Aufgabe für Experiment- und Kontrollgruppe
    - (Paare führten ferner zusätzliche Anweisungen aus, damit die gesamte Arbeitsbelastung zw. beiden Gruppen gleich ist)
  - Studenten wurden in zwei Gruppen aufgeteilt:
    - 28 in die Experimentgruppe (PP)
    - 13 in die Kontrollgruppe
    - Gleichmäßige Verteilung von guten, mittleren und schlechten Mitgliedern auf diese Gruppen

[WilKesCun00] Das Experiment: Versuchsaufbau und –durchführung (2)

- Gemessen wurden:
  - **Qualität in Anzahl der Defekte**: Nach der Entwicklung ließ ein unabhängiger Assistenten automatisierte Tests laufen.
  - **Verstrichene Zeit**: Die Studierenden mussten die auf die Lösungen verwendetet Zeiten über eine web-basiertes Applikation erfassen.
  - **Freude am Lösungsprozess**: Die Programmierer wurden hierzu zu sechs Zeitpunkten befragt.

[WilKesCun00] Das Experiment: Ergebnisse (1)

- Qualität in Anzahl der Defekte:

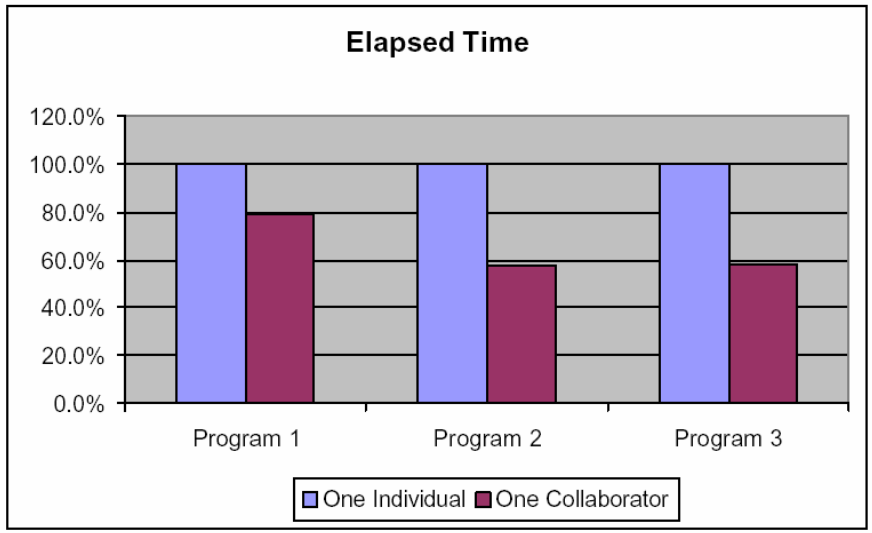
	Individuals	Collaborative Teams
Program 1	73.4%	86.4%
Program 2	78.1%	88.6%
Program 3	70.4%	87.1%
Program 4	78.1%	94.4%

Table 1: Percentage of Test Cases Passed

  - Der Code der Paare passierte immer mehr Testfälle
    - Unterschied der Qualitätsstufen ist statistisch signifikant ($p < 0{,}01$)
  - Ergebnis der Paare war konsistenter, Einzelprogrammierer variierten mehr um den Mittelwert
  - Einzelprogrammierer gaben Programme zeitweise zu spät ab, Paar taten dies nie

[WilKesCun00] Das Experiment: Ergebnisse (2)

- Verstrichene Zeit:



  - Die durchschnittlich verstrichene Zeit ist bei Paaren immer geringer.
  - Der Gesamtzeitaufwand ist bei der ersten Aufgabe noch durchschnittlich um 60% höher. Nach dieser "Einschwingphase" sinkt der Mehraufwand drastisch auf 15%.

[WilKesCun00] Das Experiment: Ergebnisse (3)

- Freude am Lösungsprozess:
  - Vorbemerkung: Zu beginn des Kurses wurden die Studenten schon initial befragt:
    - 35 (85%) gaben eine Vorliebe für PP an
  - Konstant über 90% gaben an, dass sie am Pair Programming mehr Spaß als am alleine Programmieren haben.
    - Allerdings wird im Artikel hierbei von 41 *"collaborative programmers"* gesprochen -> ???

[WilKesCun00] Das Experiment: Beurteilung

- Leider sagt der Artikel nichts zu folgenden Fragen:
  - Wie wurden die Paare gebildet?
  - Komplexität der Aufgaben
    - War unterschiedliches Spezialwissen notwendig?
  - Ist es während der Programmierung zu Informationsaustausch gekommen?
  - Wie hoch ist der Mehraufwand der Einzelprogrammierer, um zu den Paaren gleichwertige Lösungen zu erzeugen?
  - Wie würde sich das Verhältnis im Aufwand ändern, wenn von den Programmierern die Testfälle mitzuentwickeln wären?
  - In wie weit spielte Design beim Lösen der Aufgabe eine Rolle?
    - Welche Phasen wurden beim PP durchlaufen?

[WilKesCun00] Weitere Aussagen: Pair Development

Quellen?

- Zusammenfassung von Aussagen erfahrener Programmierer, welche Teile des Entwicklungszyklus in Paaren durchgeführt werden sollten (und welche nicht):
  - *"pair-analysis"* und *"pair-design"* sind kritisch für den Erfolg des Paares, denn
    - das Paar muss den in diesen Phasen festgelegten Entwicklungsrichtungen und Strategien *zusammen zustimmen*
    - es werden *mehr mögliche Lösungen* gefunden und es wird *schneller entschieden*, welche die Beste für die Implementierung ist
      - Bei der Analyse und dem Design gilt: *"two brains are better than one"*
  - Interessanterweise wird die *"pair-implementation"* von Entwicklern als weniger kritisch angesehen
    - Paare teilen sich oft bewusst für leichte oder Routineprogrammierung
      - Sie halten die Durchführung solcher Tätigkeiten alleine für effektiver
  - Das *"pair-testing"* wird als unkritischster Teil des Entwicklungszyklus gesehen
    - Dies gilt solange, wie die Paare die Testfälle zusammen entwickelt haben

[WilKesCun00] Weitere Aussagen: Pair Development

- Aus den Aussagen zum Pair Development in [WilKesCun00] könnten folgende Fragestellungen resultieren:
  - Ist "pair-programming" ohne "pair-analysis" und "pair-design" überhaupt sinnvoll/möglich?
  - Welchen Benefit bringen "pair-analysis" und "pair-design"?
  - Wie wird entschieden, ob eine bestimmte Tätigkeit besser alleine oder als Paar durchgeführt werden sollte?

Was bringt Pair Development, wie steuert man es und wie integriert es sich in vorhandene Softwareprozesse?

[KatWilWie04] Katira, Williams et al.: On Understanding Compatibility of Student Pair Programmers: Inhalt

- Es werden vier Hypothesen bzgl. der Kompatibilität von einzelnen Studenten beim PP aufgestellt und untersucht.
- Kompatibilität soll hier beschreiben:
  - Wie gut/effizient arbeiten die Partner zusammen?
  - Gemessen wurde dies durch eine Befragung.
- Die Hypothesen besagen im einzelnen, dass Paare kompatibler sind, wenn Studenten mit …
  - Hypothese 1: … unterschiedlichem Personentyp (nach Myers Briggs) gepaart werden.
    - Der *"Myers Briggs Type Indicator"* (MBTI) klassifiziert Menschen in 16 Personentypen (Temperamente).
    - Die Klassifizierung fußt auf der Bewertung folgender Eigenschaften:
      - extrovertiert (extrovert, E) oder introvertiert (introvert, I) [soziale Energiequelle]
      - wahrnehmend (sensing, S) oder intuitiv (intuitive, N) [Art der Informationsaufnahme]
      - fühlend (feeling, F) oder denkend (thinking, T) [Art der Informationsverarbeitung]
      - hinnehmend (perceiving, P) oder entscheidend (judging, J) [Art der Anwendung von Informationen und Meinungen]

[KatWilWie04] Katira, Williams et al.: On Understanding Compatibility of Student Pair Programmers: Inhalt

- Paare sind kompatibler, wenn Studenten mit …
  - Hypothese 2: …einem ähnlichen aktuellen Skill-Level gepaart werden.
    - Gemeint: Qualifikation und Erfahrung
    - Gemessen: "midterm score"
  - Hypothese 3: …ähnlich wahrgenommener technischer Kompetenz gepaart werden.
    - Gemessen durch gegenseitige Beurteilung
  - Hypothese 4: …ähnlichem Selbstwertgefühl gepaart werden.
    - Gemessen durch Selbstbeurteilung

[KatWilWie04] Versuchsaufbau und –durchführung (1)

- Versuchsaufbau und -durchführung
  - Befragungen von Studierenden unterschiedlicher Klassen der North Carolina State University (NCSU) (2002-2003)
  - Teilnehmer
    - Kurs *"CS1: Introduction to Programming (Spring 2003)"* (*freshman*)
      - 387 Studierende
      - Studierende mussten in diesem Semester 4 Projekte erledigen
      - Jede Woche ein dreistündiges "Labor" in Paaren
      - Jedes Projekt mit neuem, zugewiesenem Partner
      - 1548 potentielle Antworten bei Befragungen (erhalten: 1003)
    - Kurs *"SE: Software Engineering (Fall 2002)"* (*advanced undergraduate*)
      - 140 Studierende
      - Studierende mussten in diesem Semester 5 Aufgaben erledigen
      - Jede Woche ein zweistündiges "Labor" in Paaren
      - Für Aufgaben 1 bis 4 jeweils mit einem neuen Partner, Aufgabe 5 mit selben Partner wie Aufgabe 4
      - 560 potentielle Antworten bei Befragungen (erhalten: 496)

[KatWilWie04] Versuchsaufbau und –durchführung (2)

- Kurs *"OO: Object-Oriented Languages and Systems (Fall 2002)"* (*graduate*)
  - 62 Studierende
  - Studierende mussten in diesem Semester 3 Projekte erledigen
  - Es gab keine expliziten "Laborsessions"
  - Option in Paaren oder alleine zu arbeiten
  - 37 Studierende wählten PP für mindestens ein Projekt und wurden dann gepaart
  - Kein MBTI bei diesen Studierenden
  - 111 potentielle Antworten bei Befragungen (erhalten: 64)

[KatWilWie04] Versuchsaufbau und –durchführung (3)

- Befragungen:
  - Mittels einer web-basierten Applikation mussten die Studierenden nach jedem Projekt/jeder Aufgabe ein Reihe von Fragen beantworten, u.a.
    - Schätze die **technische Kompetenz** deines Partners im Vergleich zu deiner eigenen ab (besser, ungefähr die selbe, schwächer)
    - Schätze die **Kompatibilität** von dir und deinem Partner ab (sehr kompatibel, OK, nicht kompatibel)
  - **Selbsteinschätzung**: Die Teilnehmer des CS1 wurden nach der ersten Aufgabe zusätzlich gebeten, sich selbst auf einer Skala von 1 bis 9 einzuschätzen:
    - **1**: Ich programmiere nicht gerne und glaube, dass ich darin nicht gut bin. Ich kann einfache Programme schreiben, habe aber Probleme, neue Programme zur Lösung neuer Probleme zu schreiben.
    - **9**: Ich habe bisher keinerlei Probleme, die Programmieraufgaben fertig zu stellen, genau genommen waren diese nicht herausfordernd genug. Ich liebe es zu programmieren und rechne mit keinen Schwierigkeiten in diesem Kurs.

[KatWilWie04] Versuchsaufbau und –durchführung (4)

- Überprüfte Hypothesen bei den einzelnen Kursen:

Hypothesis: Pairs are more compatible if students with …		**CS1**	**SE**	**OO**
H-1	… different **personality types** are grouped together.	√	√	
H-2	… similar **actual skill levels** are grouped together.	√	√	√
H-3	… similar **perceived technical competence** are grouped together.	√	√	√
H-4	… similar **self-esteem** are grouped together.	√		

[KatWilWie04] Ergebnisse

- Die ermittelte Kompatibilität:

Class	N	Very Compat.	OK	Not Compat.
CS1	1003	63% (633)	26% (264)	11% (106)
SE	496	65% (324)	27% (132)	8% (40)
OO	64	72% (46)	19% (12)	9% (6)

  - Weniger als 10% der Datenpunkte signalisieren Inkompatibilität!

[KatWilWie04] Ergebnisse H-1

- H-1: Kompatibler wenn unterschiedlicher Personentyp
  - Korrelation der Kompatibilitätseinschätzungen mit dem Nicht-Zusammenpassen der Personentypen
    - Personentyp "matched", wenn alle vier Eigenschaften identisch sind

Class	N	Very Compat.	OK	Not Compat.
CS1-Matched	509	61% (307)	26% (132)	13% (70)
CS1-Unmatched	123	73% (90)	19% (23)	8% (10)
SE – Matched	259	65% (157)	27% (78)	8% (24)
SE – Unmatched	237	65% (154)	26% (61)	9% (22)

  - Bei CS1: Paare sind kompatibler, wenn unterschiedlicher Personentyp vorliegt (*"spearman rank-order correlation"* ($p < 0{,}039$))
  - Bei SE: keine Bestätigung ($p < 0{,}34$)

[KatWilWie04] Ergebnisse H-2

- H-2: Kompatibler wenn ähnlicher Skill-Level
  - Korrelation der Kompatibilitätseinschätzungen mit dem "midterm scores"
  - Bei CS1 und SE: Es konnte keine Verbindung gezeigt werden (*"spearman rank-order correlation"* ($p < 0{,}638$ bzw. $p<0{,}322$))
  - Bei OO: Starke positive Korrelation ($p < 0{,}037$) 

[KatWilWie04] Ergebnisse H-3

- H-3: Kompatibler wenn ähnliche wahrgenommene technische Kompetenz
  - Korrelation der Kompatibilitätseinschätzungen mit der Kompetenzeinschätzung
  - In allen Kursen: Signifikante positive Korrelation (*"spearman rank-order correlation"* ($p < 0.001$))

[KatWilWie04] Ergebnisse H-4 / Zusammenfassung

- H-4: Kompatibler wenn ähnliches Selbstwertgefühl
  - Korrelation der Kompatibilitätseinschätzungen mit der Ähnlichkeit der Selbsteinschätzung
  - Konnte nicht gezeigt werden
- Zusammenfassung der Ergebnisse:

	Hypothesis Pair are more compatible if students with …	**CS1**	**SE**	**OO**
H-1	… different **personality type** are grouped together	Yes	No	
H-2	… similar **actual skill level** are grouped together	No	No	Yes
H-3	… similar **perceived technical competence** are grouped together	Yes	Yes	Yes
H-4	… similar **self-esteem** are grouped together	No		

[KatWilWie04] Beurteilung

- Bei der Beurteilung des Experimentes und der Ergebnisse stellen sich folgende Fragen:
  - Was ist von der Kompatibilitätsfrage an die Paare zu halten?
    - Kompatibel bzgl. welcher (relevanter) Faktoren?
    - 90% der Paare meldeten schließlich sowieso Kompatibilität
      - Ist diese Frage vielleicht vernachlässigbar?
  - Wie gut eignet sich der "midterm score" als Maßstab für den Skill-Level?
    - Warum nur bei OO positive Korrelation zur Kompatibilität?
  - Das signifikanteste Ergebnis sagt, dass sich Studierende mit ähnlich wahrgenommener Kompetenz kompatibel fühlen
    - Allerdings hilft dies bei der Paarbildung wenig, da diese Kenntnis vorher meist nicht vorhanden!
  - Warum scheint sich der Unterschied im Personentyp nur bei den CS1-Studenten positiv auszuwirken?
- Der Artikel diskutiert diese Fragen nicht ausführlich!

Zusammenfassung

- [Nosak98]
  - Paare erzeugen lesbareren und funktionaleren Code
  - Nicht statistisch signifikant: Einzelprogrammierer sind 40% langsamer
  - Paarprogrammierer haben mehr Vertrauen in ihre Lösung und mehr Spaß am Lösungsprozess
- [WilKesCun00]
  - Nach "Einschwingphase" sinkt der Mehraufwand für PP drastisch auf 15%
  - Der Code der Paare war besser (passierte mehr Testfälle)
  - Konstant über 90% gaben an, dass sie am PP mehr Spaß als am alleine Programmieren haben
- [KatWilWie04]
  - Ähnliche gegenseitig wahrgenommene Kompetenz führt zu besserer Kompatibilität
  - Ein unterschiedlicher Personentyp spielt nur bei Anfängern ein Rolle bzgl. der Kompatibilität

Und nun?

- Sollten Untersuchungen über PP unterschiedliche Phasen des PP unterscheiden?
  - Lassen sich diese Phasen theoretisch und praktisch (z.B. beim Durchführen von diesbezüglichen Untersuchungen) gegeneinander abgrenzen? Wie?
  - Was bringt Pair Development, wie steuert man es und wie integriert es sich in vorhandene Softwareprozesse?
- Sind die Ergebnisse über die Effektivität von PP unabhängig von Randbedingungen?
- Welche Randbedingungen haben in welchen Situationen Auswirkungen? Wechselwirkungen?
  - Paarung
  - Aufgabentyp
  - Personentyp
  - Ausbildungstand /Skills
  - Projektsituation?
- Mehr Fallstudien?
- Kann es so etwas wie eine Entscheidungsmatrix geben?

Sonstige Quellen

- A. Anderson, Ralph Beattie, Kent Beck et al.: Chrysler Goes to "Extremes", 1998
- Laurie A. Williams, Robert R. Kessler: All I Really Need to Know about Pair Programming I Learned In Kindergarten, 2000

Danke