Ein Rahmenwerk für die qualitative Analyse der Paarprogrammierung

Stephan Salinger

Institut für Informatik
Freie Universitat Berlin
stephan.salinger@fu-berlin.de

Abstract: *Hintergrund:* Der Praktik der Paarprogrammierung (PP) werden verschiedene Vor- und Nachteile zugeschrieben. Der weitaus überwiegende Teil der bisherigen Untersuchungen der PP kommt allerdings bez. der Eigenschaften der PP zu keinem einheitlichen Bild. Dies liegt vor allem daran, dass in diesen meist quantitativen Studien der eigentliche PP-Prozess als Black Box betrachtet wird. *Ziel:* Mit der vorliegenden Arbeit wird angestrebt, den ersten entscheidenden Schritt zu einem holistischen Verständnis des PP-Prozesses zu erbringen und somit langfristig dabei zu helfen, die Praktik in der Industrie zielführend einsetzen zu können. *Methode:* Qualitative Analyse von Videoaufzeichnungen industrieller PP-Sitzungen mit Hilfe der Grounded Theory Methodology (GTM). *Ergebnis:* Es wird sowohl eine Forschungsmethode (Erweiterung der GTM) sowie ein von Regeln begleitetes Kategoriensystem zur Analyse des Prozesses der PP bereitgestellt. Beide Teile zusammen erlauben es, in weiteren, aufeinander aufbauenden Studien spezielle Aspekte der PP zu analysieren und so sukzessive zu einem Gesamtverständnis der PP zu kommen. Die hier vorgestellten Untersuchungen wurden 2013 im Rahmen einer gleichnamigen Dissertation veröffentlicht [Sal13].

1 Einführung

Bei der Praktik *Paarprogrammierung* (PP) teilen sich zwei Softwareentwickler einen Computer und somit in der Regel auch eine Tastatur und eine Maus, um zeitgleich zusammen Artefakte (im allgemeinen Programmkode) zu bearbeiten. Die Praktik kommt häufig, aber keineswegs ausschließlich, im Rahmen agiler Softwareentwicklungsprozesse zum Einsatz. Ihr werden unterschiedliche Vor- und Nachteile zugeschrieben. Befürworter der PP betonen folgende Eigenschaften (siehe z.B. [HDAS09]): 1. *Erhöhte Geschwindigkeit:* Programmkode kann schneller entwickelt und zum Einsatz gebracht werden; 2. *Verbesserte Qualität:* Der entwickelte Programmkode enthält weniger Defekte; 3. *Verbessertes Design:* Der entwickelte Programmkode ist besser entworfen und leichter lesbar; 4. *Konzentrierteres Arbeiten:* Paarprogrammierer arbeiten konzentrierter als Soloprogrammierer; 5. *Erhöhte „Truck Number":* Jeder Teil des Kodes ist zumindest zwei Programmierern bekannt und kann somit zumindest von zwei Personen ohne längere Einarbeitungszeit korrigiert oder weiterentwickelt werden; 6. *Verbesserter Wissenstransfer:* Die Partner lernen voneinander in unterschiedlichen Bereichen (Arbeitsstil, Design, Kodierpraktiken, Details von Programmiersprachen und Bibliotheken, Anforderungen, Anwendungsdomänen etc.);

7. *Größeres Vertrauen:* Paarprogrammierer haben ein höheres Vertrauen in ihre Arbeitsergebnisse; 8. *Höhere Zufriedenheit:* Paarprogrammierer haben mehr Spaß an ihrer Arbeit. Kritiker befürchten hingegen zumindest die beiden folgenden Effekte: 1. *Höhere Kosten:* Wenn ein Paar nicht doppelt so schnell wie ein einzelner Entwickler arbeitet, ist PP teurer als die klassische Form der Programmierung mit nur einem Entwickler pro Aufgabe (*Soloprogrammierung* (SP)); 2. *Starke Aversionen:* Einige Entwickler lehnen eine derartige Form der Zusammenarbeit prinzipiell oder in Bezug auf bestimmte Partner ab. PP mindert deren Produktivität und Motivation.

Die Annahmen über die Eigenschaften sowie das in den letzten Jahren allgemein hohe Interesse an agilen Entwicklungsmethoden haben dazu geführt, dass die PP zu einem beachteten Forschungsthema innerhalb des Software Engineerings geworden ist. Sie wurde in einer Vielzahl von Studien zumeist quantitativ analysiert. Der eigentliche Prozess wurde hierbei in der Regel als Black Box betrachtet, also nicht näher untersucht. Legt man die Ergebnisse dieser Studien nebeneinander, so ergibt sich zumeist ein heterogenes Bild: Viele der propagierten Eigenschaften werden unterschiedlich bewertet. So benötigten beispielsweise die Soloprogrammierer bei Nosek [Nos98] im Durchschnitt (nicht signifikante) 41% länger als die Paarprogrammierer, während bei Arisholm et al. [AGDS07] die Paare „nur“ 8% schneller waren. In zwei Studien zeigte sich sogar ein negativer Effekt der PP auf die Entwicklungsgeschwindigkeit, also eine Verlangsamung.[1] Darüber hinaus liefern die bisherigen Resultate weder (verifizierte) Erklärungsmodelle für die beobachteten Effekte noch für die Diskrepanzen zwischen den Ergebnissen unterschiedlicher Studien. Auch wurde bisher nicht versucht, eine allgemeine Theorie der PP zu erarbeiten oder Grundlagen für eine solche zu legen und somit die PP als zu erlernende Methode im Rahmen von Softwareentwicklungsprozessen zu etablieren.

2 Zielsetzung

Betrachtet man die bisherigen Untersuchungen zur PP, kann festgehalten werden, dass der weitaus überwiegende Teil der Arbeiten zwar Zahlen liefert, deren Zustandekommen aber nicht hinreichend erklären und somit kaum einen Beitrag zur Optimierung des Einsatzes der Praktik liefern kann. Die hier beschriebene empirische Untersuchung leistet einen ersten Schritt, um diese Lücke zu schließen. Den Ausgangspunkt bildete die Einsicht, dass ein Verständnis der Wirkungsweise der PP nur erlangt werden kann, wenn in einem ersten Schritt die interne Prozessstruktur, der sogenannte *Mikroprozess* [Ost06] der PP, analysiert wird. In einer Pilotstudie zeigte sich schnell, dass der Versuch, alle relevanten Mikroprozesseigenschaften auf einmal bzw. zusammen zu analysieren aufgrund der Vielzahl der potentiellen relevanten Aspekte zum Scheitern verurteilt ist (siehe hierzu auch [SPP08]). Das Ziel der hier beschriebenen Arbeit konnte deshalb nicht darin bestehen, eine möglichst vollständige Theorie des Prozesses der PP bereitzustellen, sondern musste erst einmal ins Auge fassen, die Grundlagen für zukünftige Erforschungen des Mikroprozesses der PP zu schaffen – und zwar derart, dass sich deren Ergebnisse direkt konsolidieren lassen und

[1] Siehe die Metaanalyse von Hannay et al. [HDAS09] zu den Faktoren Entwicklungsgeschwindigkeit, Entwicklungsaufwand/Kosten und Qualität.

dann auf längere Sicht zu einer umfassenden Theorie über die PP führen. Da hierbei nur in sehr geringem Maß auf vorhandene wissenschaftliche Erkenntnisse über den Paarprogrammierungsprozess an und für sich und somit auch nicht auf Erfahrungen in Hinsicht auf diesbezüglich geeignete Forschungsmethoden zurückgegriffen werden konnte, wurden zwei Kernziele festgelegt[2]:

1. Entwicklung einer speziell auf die Analyse des Paarprogrammierungsprozesses zugeschnittenen qualitativen Untersuchungsmethode.

2. Entwicklung eines Kategoriensystems samt Anwendungsregeln – später zusammen *Basisschicht* (BS) genannt – mit dem die den Prozess der PP formenden grundlegenden Aktivitäten der einzelnen Paarmitglieder, die sogenannten *Basisaktivitäten* der PP, beschrieben bzw. konzeptualisiert werden können. Mit der BS sollte ein Werkzeug zur Verfügung gestellt werden, das es ermöglicht, in nachgelagerten, partiell aufeinander aufbauenden Untersuchungen spezielle Aspekte des Paarprogrammierungsprozesses (z.B. den Wissenstransfer oder den Lösungsprozess) im Detail beleuchten zu können. Insbesondere sollte die BS weitere Untersuchungen dadurch erleichtern, dass bei den jeweils notwendig werdenden Konzeptionalisierungen nicht bei Null, sprich mit einer leeren Konzeptmenge, begonnen werden muss, sondern stattdessen die Möglichkeit besteht, das Kategoriensystem – später *Basiskonzeptmenge* (BKM) genannt – oder seine Elemente auszudifferenzieren, zu verallgemeinern, zu erweitern, zu beschneiden, zu ergänzen oder zum Zweck der Verbesserung der theoretischen Sensibilität [SC90] einzusetzen. Die BKM war also von vorn herein nicht als reines Kodierschema angelegt.

Entscheidend für die Herleitung war die Festlegung, dass die BS/BKM nicht unter Zuhilfenahme von allgemeinen Theorien oder anderen Kodierschemata postuliert, sondern direkt aus Sitzungsdaten extrahiert werden muss. Hierdurch sollte sichergestellt werden, dass mittels der BS/BKM nachfolgend tatsächlich auf die PP passende Theorien entwickelt werden können (siehe hierzu auch die Diskussion „Emergence vs. Forcing“ in [Kel05]). Eine Kernaufgabe der Extraktion bestand also darin, empirisch begründet zu charakterisieren, was unter einer Basisaktivität in Hinsicht auf Inhalt und Granularität verstanden werden soll. Die Entwicklung der Forschungsmethode und die Herleitung der BS gingen zumindest zu Beginn der Untersuchungen Hand in Hand.

3 Ergebnisse

Im Weiteren werden die Ergebnisse bez. der beiden oben aufgeführten Ziele erörtert. Hierbei mag es verwunderlich erscheinen, dass einer der beiden Ergebnisblöcke mit dem Begriff *Untersuchungsmethode* überschrieben ist. Hier ist zu beachten, dass es sich keineswegs nur um die Beschreibung der im Rahmen der hier vorgestellten Arbeit verwende-

[2] Es existiert zwar eine Reihe auf die Untersuchung von Programmierarbeit bezogener Kodierschemata (z.B. [ML99]), diese weisen aber jeweils zumindest eine der folgenden Eigenschaften auf und sind somit im Rahmen der angestrebten Untersuchungen nur begrenzt geeignet: i) Keine Ausrichtung auf die Analyse dialogischer Kommunikation oder die PP an und für sich. ii) Rückgriff auf Schemata aus anderen Bereichen, deren Eignung oder Angemessenheit für das hier angestrebte Ziel nicht per se offensichtlich ist. iii) Ausrichtung auf qualitativ-quantitative Studien. iv) Eingeschränkte Ausrichtung auf Teilaspekte der PP.

ten Methode handelt, sondern um die Erörterung eines Verfahrens für zukünftige, sich ergänzende bzw. aufeinander aufbauende Untersuchungen der PP.

3.1 Untersuchungsmethode

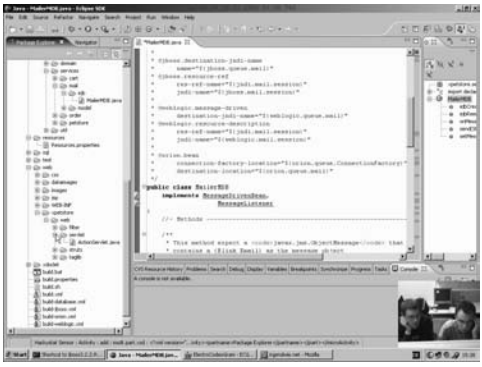

Abbildung 1: Aufbau der analysierten Videos aus drei miteinander synchronisierten Teilen: 1. Desktop des Rechners, an dem das Paar arbeitet, 2. Webcam-Video der Entwickler (unten links) und 3. während der Sitzung aufgezeichneter Ton.

Den Kern der entwickelten Analysemethode bildet die qualitative Untersuchung von Audio-/Videoaufzeichnungen von Paarprogrammierungssitzungen (siehe Abbildung 1) mittels der *Grounded Theory Methodology* (GTM) nach Strauss und Corbin [SC90][3]. Mit der Wahl der GTM, die im Kern ein exploratives Vorgehen vorsieht, wurde der Tatsache Rechnung getragen, dass zu Beginn kaum Erkenntnisse vorlagen, die sich direkt auf den Mikroprozess der PP beziehen. Der GTM wurden – nachdem erste Analyseversuche zeigten, dass die Gefahr besteht, in den komplexen Videodaten verloren zu gehen – vier neu entwickelte Praktiken zur Seite gestellt. Erste Versionen der Praktiken wurden 2007/2008 veröffentlicht [SPP08].

1. Blickwinkel auf die Daten: Durch die Verwendung eines sogenannten Blickwinkels lässt sich die bei einer GTM-Untersuchung notwendigerweise offene Forschungsfrage strukturiert formulieren. Er hilft somit dabei, während der Untersuchung interessierende besser von weniger interessierenden Phänomenen unterscheiden zu können. Über den Blickwinkel auf die Daten werden drei Aspekte festgelegt: Die Ausrichtung des Interesses, die Art der überhaupt zu berücksichtigenden Phänomene sowie der Ergebnistyp der Untersuchung (bei dem es sich nicht immer notwendigerweise um eine Theorie handeln muss). Der Blickwinkel muss explizit als etwas Vorläufiges angesehen und im Laufe einer Untersuchung regelmäßig hinterfragt werden.

Der zu Beginn der Herleitung der BS festgelegte Blickwinkel [Sal13, S. 122 ff] besagte, dass die den Prozess der PP formenden grundlegenden Aktivitäten der einzelnen Paarmitglieder zu ermitteln und zu konzeptualisieren sind, um eine extensionale Beschreibung eines auf die PP bezogenen Aktivitätsbegriffs zu erhalten. Hierbei sollten nur direkt beobachtbare Phänomene, d.h. im Wesentlichen Aspekte von tatsächlichem Verhalten berücksichtigt werden. Das Ergebnis sollte aus einer Menge von Konzepten (*Basiskonzepten*) bestehen, die weder mittels Eigenschaften ausdifferenziert noch mit Relationen versehen sein muss.

[3] Im Rahmen der Dissertation wurde hierbei die QDA-Software ATLAS.ti (http://www.atlasti.com) eingesetzt, um ohne Umweg über Transkriptionen direkt auf Videos kodieren zu können.

2. Syntaktische Regeln für Konzeptnamen: Die GTM macht keinerlei Vorgaben oder Anmerkungen dahingehend, wie Konzepte zu benennen sind. Dieser Freiheitsgrad führte im Rahmen der ersten Analyseversuche dazu, dass unterschiedlichste Formen von Konzeptnamen verwendet wurden, die sich darüber hinaus auf schwierig auseinanderzuhaltenden Granularitäts- und Abstraktionsniveaus bewegten. Diese Diversität machte es zunehmend schwierig, sich bei einzelnen Kodierungen an alle potentiell geeigneten Konzepte zu erinnern oder Konzepte bzw. Gruppen von Konzepten miteinander zu vergleichen. Hieraus wurde der Schluss gezogen, dass es – zumindest in der ersten Phase der Analyse – hilfreich sein kann, mit einem Namensschema zu arbeiten. Auf Basis der in den ersten Untersuchungen gesammelten Erfahrung wurde ein solches für die BKM festgelegt. Es besteht im Kern aus einem Verb, das die Aktivität beschreibt, der ein Entwickler gerade nachgeht, und einem Objekt, auf das sich die Aktivität bezieht (in der Form *verb_object*).

3. Modell der Analysemethoden und -objekte: In der ersten Phase der Untersuchung zeigte sich, dass drei mit der Analysemethode und dem Analysewerkzeug zusammenhängende Aspekte Probleme bereiten: i) Der dem GTM-Prozess inhärente fließende Übergang zwischen Phänomenwelt und konzeptueller Welt, ii) fehlende Leitlinien, mit denen Ideen aus der GTM systematisch zur Lösung von Kodierproblemen eingesetzt werden können sowie iii) unterschiedliche Begriffsbildungen in der GTM und der verwendeten Analysesoftware (hier ATLAS.ti). Um diesen Problemen zu begegnen, wurde das sogenannte Modell der Analysemethoden und -objekte entwickelt. Dieses Modell zeigt in Form eines UML-Klassendiagramms, welche Objekte in welcher Form auf welchem Abstraktionsniveau betrachtet werden sollten [Sal13, S. 112 ff]. Es kann jederzeit im Analyseprozess herangezogen werden, um Fragen wie „Worüber rede ich gerade?“ oder „Welche Form von Erkenntnis strebe ich gerade in welcher Weise an?“ zu beantworten.

4. Kodieren im Paar: Als neue Praktik wurde das Kodieren im Paar eingeführt. Sie fußt auf der Empfehlung von Glaser und Strauss, „die theoretischen Begriffe mit einem oder mehreren Teamkollegen zu diskutieren“, um „Verfehltes klarzustellen sowie Gesichtspunkte hinzuzufügen, auf die [die Kollegen] während der eigenen Kodierung und Datenerhebung gestoßen sind“ [GS05, S. 114], dehnt den Ratschlag aber auf alle Teile einer GTM-Analyse aus: Alle Kodierarbeit sollte zumindest zu Beginn einer Untersuchung von zwei Personen zusammen – und im Falle der Verwendung von ATLAS.ti oder einer anderen QDA-Software auch an einem Rechner – also analog der PP durchgeführt werden.

Die Verwendung der Praktiken führte dazu, dass eine Vielzahl der Probleme, die die ersten Analyseversuche behindert hatten, verschwanden oder abgemildert wurden [SPP08].

3.2 Die Basisschicht

Das Hauptergebnis der Arbeit bildet die *Basisschicht* (BS), welche aus einer Menge von Konzepten – der *Basiskonzeptmenge* (BKM) – und aus Regeln zur Anwendung dieser Konzepte besteht (ca. 300 Seiten in [Sal13]). Weiterführende Untersuchungen sollten die Regeln allerdings keineswegs stur anwenden, sondern vor dem Hintergrund neuer, sich aus der Analyse weiterer Daten ergebenden Erkenntnisse anpassen.

Tabelle 1: Nicht zusammenhängende Beispiele für die Verwendung des Konzeptes *propose_strategy*.

Äußerung	Kontext/Kommentar	Typ
„Also erst mal würd' ich den `EmailSpy` komplett außen vor lassen. Und erst mal nur den‿den `XPetstore` umstellen."	Der Entwickler teilt das weitere Vorgehen in zwei Schritte auf: Zuerst sollen die Klassen der Applikation `XPetstore` vollständig umgestellt, danach an der neuen Funktionalität `EmailSpy` gearbeitet werden.	OWP
„Weißt Du, was wir jetzt machen? Unseren vorhandenen `EmailSpy` benutzen wir zum Debuggen."	Der Sprecher schlägt eine Strategie vor, wie vorgenommene Kodeänderungen im weiteren Verlauf getestet werden können. Es soll ein Programm verwendet werden, welches vom Paar im Vorfeld der Sitzung entwickelt worden war. Dieses spricht genau die Funktionalitäten an, die momentan vom Paar editiert werden. Das Paar erspart sich hierdurch kompliziertere Testaufrufe.	DPR
„Und dann müssen wir nur noch dieses Ding überall da einbauen wo (˜sich Freunde ändern)."	Der Sprecher ergänzt einen Vorschlag des Partners um einen weiteren Arbeitsschritt. Mit der adverbialen Bestimmung „nur noch" stellt er heraus, dass man auf diese Weise zum Ziel gelangt.	EXS

Um zu einer stabilen Form[4] der BS zu kommen, wurden sieben Paarprogrammierungssitzungen mit einer Gesamtlänge von ca. 14 Stunden untersucht. Bis auf eine Sitzung, die im Rahmen einer Hauptstudiumsveranstaltung an der FU Berlin stattfand, handelte es sich bei allen analysierten Sitzungen um solche mit professionellen, PP-erfahrenen Entwicklern. Sie wurden in drei verschiedenen Firmen aufgezeichnet. Es wurde kein Einfluss auf die Auswahl der zu bearbeitenden Programmieraufgabe genommen. Diese stammten vielmehr direkt aus dem jeweiligen Arbeitsalltag (Java und PHP). Die aufgezeichneten Sitzungen hatten Längen zwischen gut einer und knapp drei Stunden. Drei der Sitzungen wurden soweit im Detail untersucht, dass weitgehend alle Äußerungen und Handlungen kodiert wurden (letztendlich 3008 Kodeinstanzen an 2464 Videoausschnitten). Die Anwendung der erweiterten GTM führte dazu, dass zwei Gruppen von Konzepten identifiziert wurden:

1. HHI (human-human interaction): Die Klasse HHI umfasst alle Konzepte, mit denen Interaktionen zwischen den Paarmitgliedern, also aufeinander bezogene Handlungen, beschrieben werden können. Da verbale Kommunikation als zentrale Interaktion identifiziert wurde, wurde festgelegt, dass die Klasse nur solche Elemente enthalten soll, die der Konzeptualisierung verbaler Äußerungen dienen. Nachfolgende Untersuchungen sollten aber darüber nachdenken, das Anwendungsgebiet der Klasse HHI auszudehnen.

2. HCI/HEI (human-computer interaction/human-environment interaction): Die Menge HCI umfasst alle Konzepte, die Entwickleraktivitäten adressieren, die sich auf den Computer bzw. auf Programme, die auf ihm laufen, beziehen, während die Klasse HEI alle Konzepte bündelt, die sich auf Aktivitäten beziehen, die in Hinsicht auf die sonstige Arbeitsumgebung geschehen.

Im Wesentlichen wurden 69 Konzepte in den Daten gegründet, von denen die HHI-Konzepte den Kern der BKM bilden. Sie verteilen sich auf vier Hauptklassen:

Produktorientierte Konzepte werden im Zusammenhang mit Vorschlägen (und Entgeg-

[4] Eine erste Version der BS wurde 2008 auf dem Workshop der Psychology of Programming Interest Group (PPIG) präsentiert [SP08].

nungen auf solche) verwendet, die die Gestalt des zu entwickelnden Programms betreffen, also die Inhalte, die Struktur und die Anordnung der Programmartefakte.

Prozessorientierte Konzepte referenzieren Vorschläge (und Entgegnungen auf solche), die sich auf die Gestaltung des Arbeitsprozesses beziehen. Z.B. adressiert die Klasse *strategy* Äußerungen zu einem längerfristig ausgerichteten, planvollen und in der Regel aus mehreren komplexeren Einzelschritten bestehenden Handeln.

Universelle Konzepte adressieren Äußerungen, mit denen Wissen angefordert, transferiert und beurteilt wird sowie Wissenslücken offenbart oder Vermutungen bzw. Hypothesen aufgestellt und bewertet werden. Sie heißen universell, da sie sowohl im Produkt- wie auch im Prozesskontext verwendet werden können.

Fassadenkonzepte bilden eine Teilmenge der universellen Konzepte und liefern einen vereinfachten, auf einen bestimmten Aspekt fokussierten Blick auf in der Regel komplexere Phänomene, z.B. auf Äußerungen oder Sequenzen von Äußerungen bez. momentan ablaufender HCI- oder HEI-Aktivitäten. Hierdurch stellen sie unter anderem einen Kontext zur Verfügung, in dem diese komplexeren Phänomene aufgesplittet und durch spezifischere Konzepte beschrieben werden können.

In Abbildung 2 sind die HHI-Konzepte in Übersicht dargestellt. Sie wurden derart konzipiert, dass bei ihrer Anwendung die folgenden Ziele verfolgt werden können: 1. Erfassen der primären Intention des Sprechenden. 2. Sichtbarmachen von sprachlichen Bezügen, beispielsweise von Frage-Antwort-Paaren oder Entscheidungsfindungsprozessen. Im Rahmen der Herleitung der BS musste deshalb herausgearbeitet werden, was unter der primären Intention einzelner Äußerungen verstanden werden soll. Anschließende Vergleiche mit Konzepten aus der Linguistik bzw. der linguistischen Gesprächsanalyse zeigten, dass das Intentionsverständnis der BS eng mit Konzepten aus der Sprechakttheorie [Sea69], z.B. dem primären und sekundären illokutionären Akt, verwandt ist.

Im Zusammenhang mit der Beschreibung der einzelnen Unterklassen – z.B. der durch das Objekt *strategy* gebildeten – wurden die sogenannten *primär charakterisierenden Eigenschaften* eingeführt. Grob gesprochen handelt es sich bei diesen um Eigenschaften, deren Kenntnis unverzichtbar dafür ist, dass die Konzepte der Klasse im intendierten, aus den Daten ermittelten Sinn verstanden und verwendet werden können. Im Falle von *strategy* gehört beispielsweise die Information darüber dazu, dass es drei, sich in vieler Hinsicht stark unterscheidende, grundsätzliche Typen von Strategien gibt, die unter dem Begriff *strategy* subsummiert werden: 1. *OWP (organising work packages)*: Teile der vorliegenden Aufgabe werden in einzelne konkrete Arbeitsschritte zerlegt; 2. *DPR (determining procedure rules)*: Es werden Leitlinien vorgeschlagen, die zumeist in einer kontinuierlichen Art und Weise helfen sollen, nachfolgende Arbeiten durchzuführen oder zu strukturieren; 3. *EXS (expansioning step)*: Taktischen Äußerungen (vom Typ *step* oder *design*) werden zu einer Strategie ausgebaut. Man beachte hierzu auch die Beispiele in Tabelle 1.

Neben den primär charakterisierenden Eigenschaften wird auch auf spezielle Eigenschaften verbaler Kommunikation eingegangen. Hiermit wird der Tatsache Rechnung getragen, dass es mittels verbaler Äußerungen vielerlei Möglichkeiten gibt, um ein und denselben Aspekt zu artikulieren, dass also gleichartige Phänomene in gänzlich unterschiedlicher Gestalt auftreten können. Außerdem wird diskutiert, wie damit umzugehen ist, dass kon-

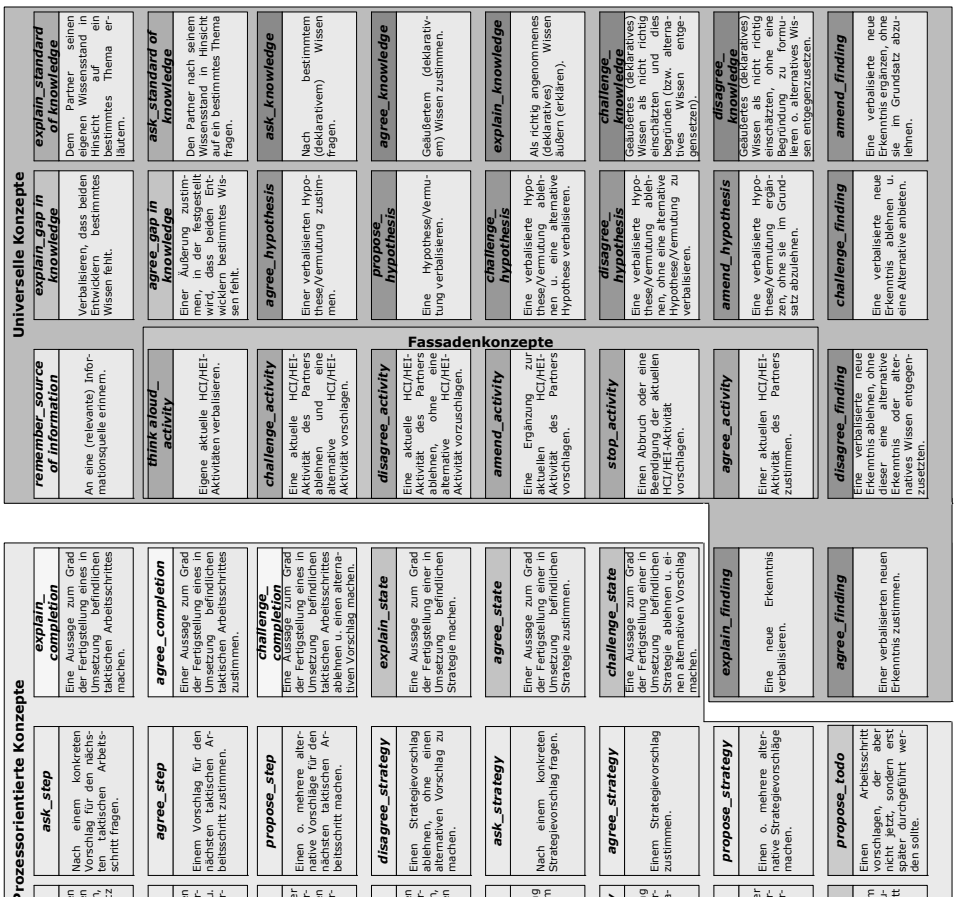

Abbildung 2: Die HHI-Konzepte geordnet nach Haupt- und Unterklassen. Einen Sonderfall bildet die Klasse *activity*. Hier beziehen sich die reaktiven Verben (z.B. *challenge*) nicht auf Äußerungen zu Äußerungen, sondern auf Äußerungen zu HCI/HEI-Aktivitäten.

textfreie oder situationsenthobene Äußerungen „im Grunde“ pragmatisch mehrdeutig sind [LNP04], der Kontext aber oft nur begrenzt eruiert werden kann. Nicht zuletzt wird besonderen Wert auf die Abgrenzung von Konzepten zueinander gelegt, und das, obwohl in der Arbeit der Ansicht gefolgt wird, dass „Skepsis einem Denken gegenüber angebracht [ist], das ‚die Welt im Rahmen eindeutig unterschiedener Kategorien‘ zu begreifen versucht“ [Muc07]. Dass trotzdem angestrebt wird, zu klaren Abgrenzungen zu kommen, geschieht aus der Überlegung heraus, dass nur so deutlich gemacht werden kann, wo dies schwierig ist und in nachfolgenden Untersuchungen genauer hingesehen werden muss.

4 Zusammenfassung und Ausblick

Mit der BS (und der Untersuchungsmethode) steht ein interdisziplinär ausgerichtetes Rahmenwerk zur Verfügung, mit dem direkt in die qualitative Analyse spezifischer PP-Aspekte eingestiegen werden kann. Es ist dahingehend konzipiert worden, die Ergebnisse unterschiedlicher PP-Studien leicht konsolidieren zu können. Die umfassende Beschreibung der BS stellt hierzu neben den eigentlichen Konzepterläuterungen/Beispielen auch eine große Anzahl von ggf. gerichtet zu modifizierenden Kodierregeln zur Verfügung [Sal13]. Um das Rahmenwerk einem größeren Kreis als Werkzeug zugänglich zu machen, wurde im Dezember 2013 ein englischsprachiges Buch zur praktischen Handhabung der BS veröffentlicht [SP13]. Ihre Feuerprobe hat die BS bereits hinter sich gebracht, z.B. beim erfolgreichen Einsatz in Untersuchungen zu den Themen Aktivitätsbeeinflussung [Zie12], Rollen (Revidierung des bisher angenommenen Driver/Observer-Rollenmodells) [SZP13] und Wissenstransfer (von Zieris und Prechelt an der Freien Universität Berlin; Veröffentlichung noch ausstehend). Sie bildet somit den Ausgangspunkt, um zu einem kontextsensitiven, prozess- und zielorientierten Verständnis und Einsatz der PP zu kommen. Nebenbei demonstriert die Arbeit, wie auch im Software Engineering gewinnbringend auf die in anderen Disziplinen häufig verwendeten qualitativen Methoden zurückgegriffen werden kann.

Literatur

[AGDS07] Erik Arisholm, Hans Gallis, Tore Dybå und Dag I.K. Sjøberg. Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *IEEE Transactions on Software Engineering*, 33(2):65–86, 2007.

[GS05] Barney G. Glaser und Anselm L. Strauss. *Grounded Theory: Strategien qualitativer Forschung*. Hans Huber, Bern, 2.. Auflage, 2005. Original erschienen 1967: The Discovery of Grounded Theory: Strategies for Qualitative Research.

[HDAS09] Jo E. Hannay, Tore Dybå, Erik Arisholm und Dag I.K. Sjøberg. The effectiveness of pair programming: A meta-analysis. *Information and Software Technology*, 51(7):1110–1122, 2009.

[Kel05] Udo Kelle. “Emergence” vs. “Forcing” of Empirical Data? A Crucial Problem of “Grounded Theory” Reconsidered. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, 6(2), 2005.

[LNP04] Angelika Linke, Markus Nussbaumer und Paul R. Portmann. *Studienbuch Linguistik*. Niemeyer, Tübingen, 5. Auflage, 2004.

[ML99] Anneliese von Mayrhauser und Stephen Lang. A Coding Scheme to Support Systematic Analysis of Software Comprehension. *IEEE Transactions on Software Engineering*, 25(4):526–540, 1999.

[Muc07] Petra Muckel. Die Entwicklung von Kategorien mit der Methode der Grounded Theory. *Historical Social Research, Supplement*, (19):211–231, 2007.

[Nos98] John T. Nosek. The case for collaborative programming. *Communications of the ACM*, 41(3):105–108, 1998.

[Ost06] Leon J. Osterweil. Unifying Microprocess and Macroprocess Research. In *Unifying the Software Process Spectrum*, Jgg. 3840 of *Lecture Notes in Computer Science*, Seiten 68–74. Springer Berlin / Heidelberg, 2006.

[Sal13] Stephan Salinger. *Ein Rahmenwerk für die qualitative Analyse der Paarprogrammierung*. Dissertation, Freie Universität Berlin, 2013.

[SC90] Anselm L. Strauss und Juliet Corbin. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage Publications, Inc., London, 1990.

[Sea69] John Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[SP08] Stephan Salinger und Lutz Prechelt. What happens during Pair Programming? In *Proceedings of the 20th Annual Workshop of the Psychology of Programming Interest Group (PPIG '08)*, Lancaster, England, September 2008.

[SP13] Stephan Salinger und Lutz Prechelt. *Understanding Pair Programming: The Base Layer*. Books on Demand, 2013.

[SPP08] Stephan Salinger, Laura Plonka und Lutz Prechelt. A Coding Scheme Development Methodology Using Grounded Theory for Qualitative Analysis of Pair Programming. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*, 4:9–25, 2008.

[SZP13] Stephan Salinger, Franz Zieris und Lutz Prechelt. Liberating Pair Programming Research from the Oppressive Driver/Observer Regime. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, Seiten 1201–1204, 2013.

[Zie12] Franz Zieris. Qualitative Untersuchungen von Beeinflussungen der Aktivität des Partners bei der Paarprogrammierung. Masterarbeit, Freie Universität Berlin, 2012.

Dr. Stephan Salinger wurde 1965 in Berlin geboren. Er studierte Mathematik und Informatik an der Freien Universität Berlin. Nachfolgend war er in verschiedenen Softwareeentwicklungs- und Consultingunternehmen als Softwareentwickler, Projektmanager und Entwicklungsleiter tätig, vornehmlich im Bereich Content Management. Als Mitglied der AG Software Engineering forscht und lehrt er seit 2003 am Lehrstuhl von Prof. Dr. Lutz Prechelt an der Freien Universität Berlin im Bereich der agilen (verteilten) Softwareprozesse. Er promovierte im Jahr 2013.