PP-ind: A Repository of Industrial Pair Programming Session Recordings

Franz Zieris
zieris@inf.fu-berlin.de
Freie Universtität Berlin
Berlin, Germany

Lutz Prechelt
prechelt@inf.fu-berlin.de
Freie Universtität Berlin
Berlin, Germany

***Abstract*—PP-ind is a repository of audio-video-recordings of industrial pair programming sessions. Since 2007, our research group has collected data in 13 companies. A total of 57 developers worked together (mostly in groups of two, but also three or four) in 67 sessions with a mean length of 1:35 hours. In this report, we describe how we collected the data and provide summaries and characterizations of the sessions.**

Contents

1. Introduction

Pair programming (PP) is a software development practice in which two developers work closely together on a technical task on the same computer. It was popularized by Kent Beck who sees it as the central practice of eXtreme Programming and describes it as *"a dialog between to people trying to simultaneously program (and analyze and design and test) and understand together how to program better"* [2, p. 100].

Controlled experiments on pair programming have shown mere tendencies in terms of effects on quality and effort with much variation left to be explained [3]. In the words of the authors of a large experiment with almost 300 hired consultants: *"we are still far from being able to explain why we observe the given effects"* [1].

Our research group has been collecting industrial pair programming sessions since 2007. We record pair programming as it happens "in the wild" in order to understand how it actually works and what really matters in everyday practice. In particular, we record the pairs' converstation, their screen content, and a webcam video showing their gestures and posture.

This kind of data data is amendable to different types of analyses. We describe our qualitative approach in [10, 11]. In this report, we describe the technicalities of how we collected the data. Several researchers have contributed plenty of time to collecting and processing that data, and we want to give credit where credit is due. The raw data itself cannot be released to the public because of non-disclosure agreements with the respective companies. As a proxy, we here characterize the companies, the developers, and their PP sessions.

This report is structured as follows: We discuss our fundamental approach to collection empirical data on pair programming (Section 2) and describe our generic data collection protocol (Section 3). We introduce some terminology and describe the structure of our data (Section 4). We give an overview of our repository (Section 5) and then discuss the individual contexts and cases in (Section 6). We close with a discussion of the properties and limitations of our data collection (Section 7) and give of an overview of which data has been used in which publications so far (Section 8). In Appendix A, we explain the technical details of how we record and process PP sessions; Appendix B is a (redacted) reprint of a report we handed out to one of the companies.

2. Fundamental Considerations

There are two fundamental considerations to our data collection: First, we record pair programming as it happens in industry. Second, we consider the pair programming session as the basic unit. Both considerations were driven by our research interests.

2.1. Naturalistic Industrial Setting

Our research has a focus on practical relevance. This why we study industrial settings with *professional software developers* working on their *everyday tasks*. This also entails that the developers work in their normal development environment, with partners they chose to work with, at times and to an extent they decide.

We primarily rely on *observation* of developers working in pairs, as opposed to reports of pair programming in, say, interviews. To enable a thorough analysis later on, we *record* the pair programmers. In particular, the pair members' interactions with one another and their computer(s) as well as the contents of their screen(s) need to be captured in audio and video. The necessary recording infrastructure somewhat reduces the naturalism of the observed session, the effects of which we discuss in Section 7.2.

2.2. The Pair Programming Session as a Unit

Our data collection starts a point where the developers already made the decision to work as a pair. Their decision, just as the project they work in, the task(s) they work on, their software system, and their team structure all may "echo" in their session and are therefore helpful for understanding their activities, but are not prime concerns of our data collection.

This constrasts, for example, with the work of Socha et al. [15, 16] who recorded 11 days of software development with a multitude of video cameras and audio recorders resulting in six terabytes of data consisting of thousands of photographs, 292 hours of screen captures, and 380 hours of video of developer interactions.

3. Data Collection Protocol

In our research group, Stephan Salinger and Laura Plonka initiated our industrial data collection efforts and they devised a protocol that served as the basis for collecting data in all companies.

The data collection protocol is *generic* in two ways. First, it is adapted in each particular installment at a company on-site to deal with constraints, to seize opportunities, and to fit the particular research focus of the researcher (see Section 3.3). Second, the protocol is still more or less independent from any particular research question regarding pair programming, as the resulting data can be reused for different purposes (some conditions apply, which we discuss in Section 7).

3.1. Protocol Overview

After a company has been approached and probed whether the company would be open to have some of their programming sessions recorded, the overall research goal, the procedure, extent, and purpose of the main data collection are explained in a presentation for the development team(s). It is also made clear to the potential subjects that all participation is voluntary and that their agreement to be recorded can be revoked at any point during a session. The heart of the data collection protocol is a cycle per session recording:

- After a pair announces that it is willing to have their next pair session recorded, the recording infrastructure is set up and the **session recording** is started once the developers are ready (see Section 3.2 for details).
- Optionally, both developers fill out **questionnaires** before and/or after their session, in which the developers state their names, development and pair programming experience, characterize the nature of their task, and whether it went as they intended (see Fig. 2).
- Afterwards, the researcher does a **quick analysis** of the material during which she looks for peculiarities that catch her attention. The main purpose of this step is to inform the next activity.
- The researcher then conducts a **reflective interview** with the developers on the day after the recording. This activity serves different purposes, including collecting background information and providing developers with feedback in return for them agreeing to have their PP session recorded and scrutinized. These interviews are audio-recorded.

3.2. Recording Sessions

The software developers themselves decide when and for how long they want to have their work recorded. They work on their own machines, in their normal environment, on their everyday tasks, and with the partner they chose.

The session recordings as technical artifacts consist of a screencast of the pair's monitor(s), the pair's conversation as audio, and a webcam video showing the two pair members' interaction. These three sources are combined to a self-contained video file as illustrated in Fig. 1. Both webcam feed and screencast are captured at between 5 and 15 frames per second (depending on what the hardware can handle), which does not make for a great movie experience but allows to distinguish individual keystrokes, to follow mouse movements, and the see the developers' gestures. The final videos' resolution depends on the developers' display(s) and recording setup and ranges from 1024×768 to 2560×1440 pixels.

The recording process relies on one of three generations of hardware and software components. The general setup works like this: The developers work on one machine, and screencast and webcam feed are transmitted to another machine where they are recorded. We explain the details in Appendix A. The most relevant difference is that generation 1 is an unattended offline recording which the researcher only gets to see once the pair is done, while generations 2 and 3 are an online recording which allows the researcher to watch the session live (and start her quick analysis) while it is still being recorded.

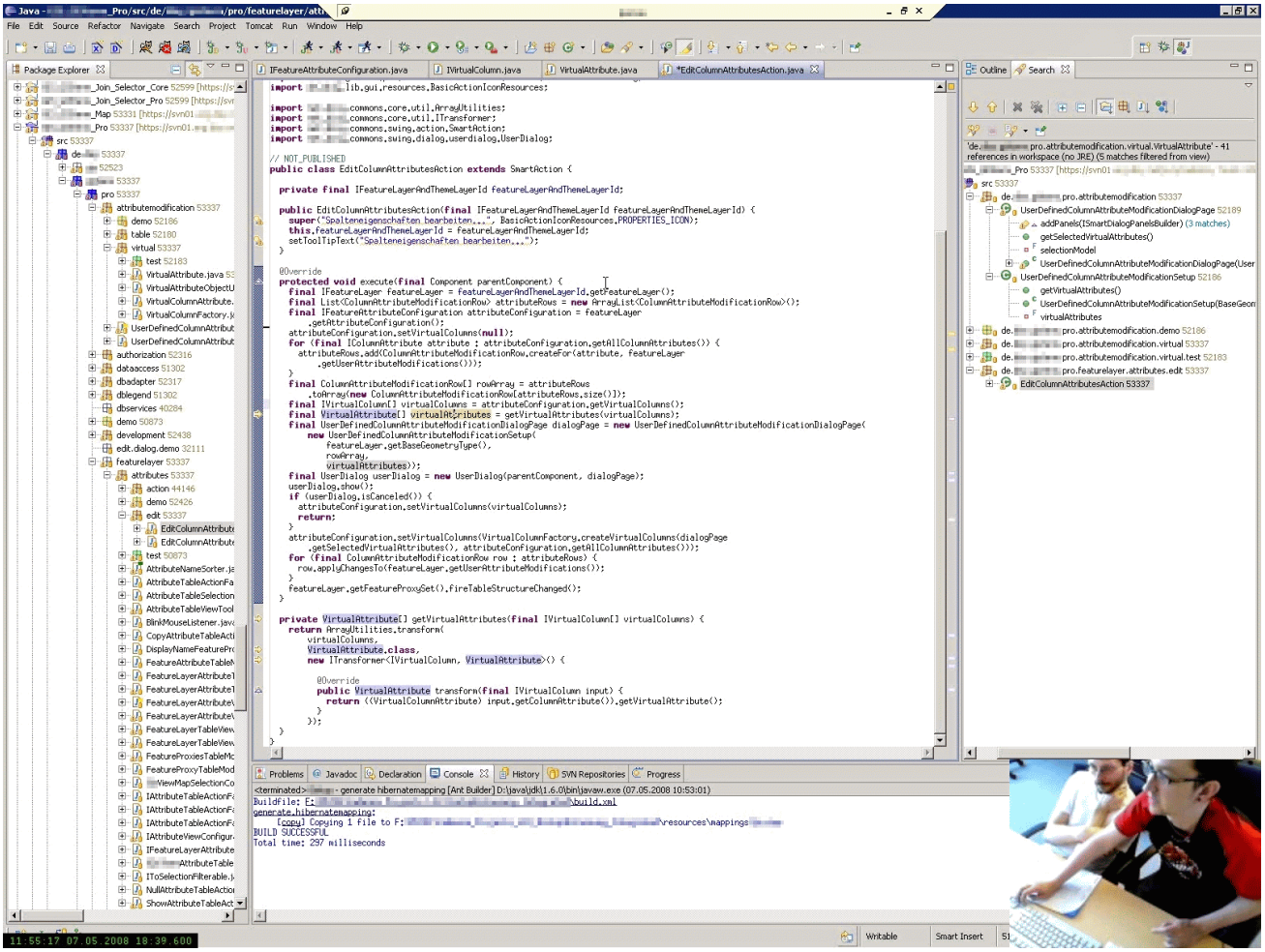


Figure 1: Still frame of a session recording (here: session **CA2**) with screencast in the background and the webcam video layered on top.

3.3. Per-Company Differences

With each installment of the data collection protocol at a new company, there were slightly different sets of mutual expectations which resulted from prior discussions with the partners and from evolved research interests. They can be grouped into three areas as to what the "headline" of the cooperation was. We discuss this difference because it affects the researcher's emphasis she puts on each of the data collection activities on-site. Additionally, the subjects' expectations and possibly also behavior are shaped by knowing *why* a researcher is around. Table 1 gives an overview of the contexts, involved researchers, and headlines.

- **PP**: Most companies were specifically approached with the intention to understand pair programming.
  While for the first contacts there was no particular focus yet, Plonka's [4] contacts to companies C to F were influenced by a particular interest in the *driver and navigator* roles, which is why the webcams in these sessions are recorded in an angle that shows the developers hands but occassionally cuts off their heads.
  Additionally, the data collection period was branded to these companies as a "workshop" to help developers reflect on their PP process. Here, one work station was setup for all pairs to use and the developers would put their names on a list to choose either a morning or an afternoon slot to be recorded for a planned maximum duration of 1.5 to 2 hours. Nevertheless, once a recording was started, the developers (just as in all other companies) were left to work on tasks of their choosing for as long they wanted.
  Schenk [13] was particularly interested in distributed pair programming and chose her contacts J and L accordingly; Zieris [17, 18] is interested in knowledge transfer and all participants from companies K, M, O, and P knew that.
- **Agile**: Salinger and Zieris approached companies K and O in an effort to understand agile software development in general. According data from these contexts was collected and analyzed (see [20], which is about company K). All particular PP sessions, however, were recorded exclusively for the purpose of understanding knowledge transfer in pair programming.
- **Onboarding**: Schmeisky, Salinger, and Zieris approached company N for understanding their onboarding process, i.e., how new-hires are integrated into the company. Pair programming was not a designated part of that process, but a number of developers agreed to be recorded while working in pairs. The recordings were therefore a window into the actual onboarding

process, and were not made to understand pair programming.

Another effect of these different headlines and focuses can be seen in the different versions of the pre- and post-session questionnaires, see Figure 2.

4. Terminology and Structure

We use the following nomenclature:

- Companies are represented by single letters: A, B, C, and so on.
- Sessions are grouped by their technical context (to distinguish different sets of requirements and/or differences in technology stacks) and then counted up with Arabic numbers. **CB1** is the first recording in the second context in the third company.
- Developers are identified through their company and Arabic numbers, such as **C3**.

4.1. Pair Programming Modes

Although our primary research focus is on pair programming, we did not restrict our recordings to a fixed settings of two developers working on the same machine. We distinguish three dimensions along which the collaboration modes in our repository differ.

- **Number of Developers**: Just as one developer may ask a colleague for help, a pair may do the same. Some teams even form groups of three or more people intentionally. So far, we recorded sessions with groups of two to four developers.
- **Spatial Distribution**: The developers can all be co-located, or all be in separate locations. (There may also be co-located subgroups, but we did not record such settings yet.)
- **Number of Active Computers**: The developers can share access to a single machine, or each of them can use their own machine, or anything in between.

In principle, these three dimensions are independent from each other. However, we did not observe every possible constellation yet. We characterize the ones we did see as follows (refer to Table 2 for a summary):

- **Pair Programming** (**PP**): Two developers sit next to each other and work on one single machine.
- **Pair Programming with Active Observer** (**PPao**): Like PP, but one partner occasionally looks up things on her own machine. The pair still works on one task though.
- **Side-by-Side Programming** (**SbS**): Two developers sit next to each other and work on their own separated but related tasks.
- **Mob Programming** (**Mob**): Three or more developers sit close to each other (e.g. in a conference room or next to each other in a row) and work on anything between one single machine, and one machine per developer. We have observed neither fully nor partially distributed Mob settings.
- **Remote Pair Programming** (**RPP**): Two developers are distributed to two different locations and share one screen, i.e., there is one developer who "owns" the machine, and the other developer may have view-only access (screen-sharing, e.g., via Skype) or some interaction options (e.g., via TeamViewer).
- **Distributed Pair Programming** (**DPP**): Two developers are distributed over two different locations and they each can interact with their development environment independently but their actions are synchronized.

4.2. Structured Developer Information

For all 57 developers, we provide the following structured information:

- **Gender** (**Gnd**): We did not ask any participant for their gender specifically. We list them as either "female" or "male" depending on the first name they told us.
- **Spoken Languages** (**Lang**): We list the developers' spoken languages in decreasing proficiency, using ISO 639-1 codes (e.g., "EN" for English, "DE" for German). We only list languages that are actually spoken in our data; a dash "–" is used as a placeholder, e.g., when it is clear that the session language is not the developer's first language and she nevers uses that first language.
- **Software Development Experience** (**Dev.**): In some companies, we asked the developers to self-report their professional software development experience in years and months in the questionnaires (see Fig. 2). In other cases, we retrospectively searched for public profiles on professional network sites.
  Pair Programming Expierence (**PP**): Here, we only relied on self-reported data from the questionnaires.
  Time with Company (**Comp.**): This aspect we did not ask for in any questionnaire. The according numbers stem exclusively from the participants' public profiles on career sites.
  For all the experience-related information we use the following conventions: A "*n.a.*" in a table cell means that no information whatsoever is available. A blank cell means that the developer did fill out the according questionnaire, but left the field blank. Normal numbers are represented as such, while special markings and comments are given in quotation marks, e.g. "ca. 6?" or "first time". If the same developer was asked the same question on multiple occasions and gave different answers, the respective values are separated by a slash "/".
- **Number of Sessions** (**#S**): The number of session recordings in our repository the developer is part of.
- **Number of Pairs** (**#P**): The number of distinct pair (or group) constellations in which the developer was recorded.

4.3. Structured Session Information

For all 67 sessions, we provide the following structured information:

- **Mode**: A characterization of how the developers work together (see Section 4.1). We list the predominant mode of the session. See the tables' respective captions and Section 6 for more details on some corner cases .

Company			Data Collection			
ID	Application Domain	Country	Year	Researchers	Headline	Focus
A	Content Management System	Germany	2007	Salinger	PP	–
B	Social Media	Germany	2007	Salinger & Plonka	PP	–
C	Geo-Information System	Germany	2008	Plonka	PP	workshop, roles
D	Customer Relationship Management	Germany	2008	Plonka	PP	workshop, roles
E	Logistics and Routing	Germany	2008	Plonka	PP	workshop, roles
F	Email Marketing	Germany	2008	Plonka	PP	workshop, roles
J	Data Management for Public Radio Broadcast	Germany	2013	Schenk	PP	distributed PP
K	Estate Online Platform	Germany	2013	Salinger, Zieris	Agile & PP	knowledge transfer
L	Freelance Training Consultant	USA	2014	Schenk	PP	distributed PP
M	Data Analysis in Energy and Transportation	Norway	2014	Zieris	PP	knowledge transfer
N	Online Fashion Retailer	Germany	2016	Salinger, Schmeisky, Zieris	Onboarding	–
O	Online Project Planning	Germany	2016	Zieris	Agile & PP	knowledge transfer
P	Online Car Part Resale	Germany	2018	Zieris	PP	knowledge transfer

Table 1: Contexts for sessions recorded by member of our research group. The research direction ("headline" and "focus") was set by the named researchers. Sessions in three additional industrial contexts (G, H, and I) were recorded with low technical quality and by students with little oversight, so important context information is missing. We do not include the according sessions in our repository PP-ind because of their low quality.

Task & Pair Items ("Pre-Session")

1) Task classification (new functionality, extend functionality, test cases, debugging, refactoring, or other)
2) Short description of the task
3) Characterization of (expected) difficulties
4) Estimated time to completion
   [added in version 3]
5) Why task is worth to be worked on by a pair
6) Professional software development experience
7) Pair programming experience
   [added in version 2]
8) How well attuned to their respective partner?
9) Expectations towards the reflective interview later on
   [added in version 2]

Process Items ("Post-Session")

1) School grade for recent session
2) Compare progress with expectations
3) Divide session into phases
4) Name most important phases
5) Assess session-specific importance of each: knowledge transfer, developing a strategy, bug fixing, developing a design/an architecture, developing an algorithm, knowing an API, having the right idea
   [removed in version 2]
6) Points where pair constellation should have been given up
7) Points where pair constellation was especially beneficial [added in version 2]

Figure 2: Shortened items from the pre-session and the post-session questionnaire which the developers filled out individually (except for the D-pairs who all handed in just one collective pre-questionnaire). There were three different versions: Version 1 was used for companies A and B, version 2 for C, and version 3 for D, E, and F (see [9, Appendix E] for the original version 1 in German and [4, Appendix A] for version 3 translated into English). Developers from J and K received and answered the *task & pair items* (except 4, 6, and 9) via e-mail *after* their respective sessions. In the other installments (M, N, O, P), no questionnaires were used. Instead, the researchers asked the developers directly in the reflective interviews.

Name	Short	#Dev	Spatial Setting	Active Computers
Pair Programming	PP	2	co-located	1
Pair Programming with Active Observer	PPao	2	co-located	1–2
Side-by-Side Programming	SbS	2	co-located	2
Mob Programming	Mob	3+	co-located	1+
Remote Pair Programming	RPP	2	distributed	1
Distributed Pair Programming	DPP	2	distributed	2

Table 2: Collaboration modes in our data collection

- **Developers**: An incident matrix of the developers participating in the session.
- **Start**: Date and time when the session proper started. For some recordings, the video starts several minutes earlier.
- **Duration** (**Dur.**): Gross length from the start to the point when the pair/group dissolves their session, including any short breaks. An intermittent stand-up meeting of 15 minutes or less counts as short whereas a lunch break does not in which case we consider the parts to be individual sessions even if they where recorded in one sitting.
- **Spoken language(s)** (**SL**): The natural languages used by the participants in the session, using ISO 639-1 codes (e.g., "EN" for English, "DE" for German).
- **Programming Language(s)** (**PL**): The structured languages of the source code files/snippets the developers read and/or write in their session.
- **Pre-Session Pair Assessment** (**Pre**): Answer to *task & pair item* 8: 'How well attuned to your partner are you?' expressed in German school grades from 1 (best) to 6 (worst).
  Post-Session Assessment (**Post**): Answer to *process item* 1: 'How useful was it to solve the task as a pair?' expressed in German school grades from 1 (best) to 6 (worst).
  For multiple session recording in one setting, only the first has Pre data and only the last has Post data. In the tables to follow, Pre and Post columns are omitted for companies in which no questionnaires were used.

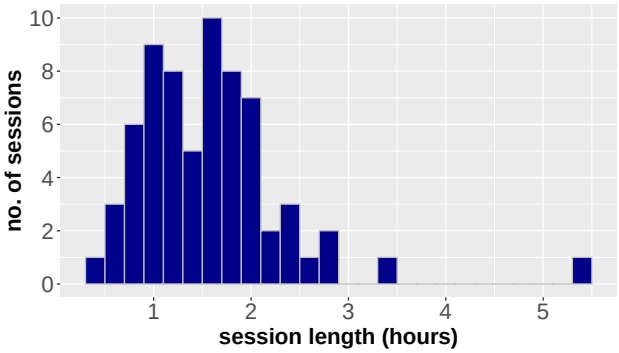


Figure 3: Distribution of Session Lengths

5. Overview of Sessions

Table 3 summarizes the contents of the repository per company. In addition to the types and lengths of the recorded sessions, we also list the quality of the auxiliary data sources, i.e., pre- and post-session questionnaires and recordings of the reflective interviews.

Figure 3 is an histogram of the sessions' respective lengths. The vast majority of our sessions is runs between 0:45 and 2:30 hours.

Attached to this arXiv publication is an R file that contains all the structured information in the form described in Sections 4.2 and 4.3.

6. The Repository

In this Section, we characterize each of the 13 companies and provide summaries for many of the individual sessions. Analyzing such sessions on a content-level is hard work and takes time. Hence, we only provide summaries for the sessions whose technical contents we already understand good enough.

6.1. Company A

Company A develops a web-based content management system (CMS) in Java and Objective-C.

We recorded a single session (**AA1**) with developers A1 and A2 in this company. Both developers know their domain well and are generally experienced developers. Developer A1 has very good structural knowledge of the Java frontend and its individual classes, as well as practical knowledge of the Eclipse IDE and the Java programming language. His colleague A2 is more familiar with the backend and the SQL database, the VIM editor and the UNIX shell, and the Objective-C programming language. However, these differences are small: Each of them would be able to find their way around in both parts of the system.

See Table 4 for general information on the developers and Table 17 for structured information on the recorded session.

Session AA1 The pair works on fixing inconsistencies across different list views. They work both in the frontend and the backend code. Although they know their code base well, they still spend some time along to way to understand its peculiarities. At the end of the session they worked through five different lists and made them consistent, but learned that one aspect of the business model cannot be implemented in the backend alone. They finish with a architectural discussion.

6.2. Company B

Company B develops a social media platform in PHP and JavaScript. We visited the company two times to record a total of four PP sessions.

All four sessions are with the same pair of full-stack developers B1 and B2. See also Salinger's description of the data collection [9, pp. 95–99]. See Table 5 for general information on the developers and Table 18 for structured information on the recorded sessions.

Note on Naming Scheme: We originally considered the data from the first recording sitting to be corrupt but were eventually able to mostly recover them. This is why the earlier sessions are labeled **BB1** to **BB3** even though they take place five months prior to session **BA1**.

Sessions BB1, BB2, and BB3 The pair implements a new feature from scratch in the course of one afternoon (half past two until half past eight), going through their complete web development stack, starting with template and internationalization in session **BB1**, continue with controller, model, database layer, and template optics in session **BB2**, and conclude with making their view more interactive through JavaScript in session **BB3**.

Note on Data: The webcam and audio recording is jumbled for the last 30 minutes of session **BB3**: While the screen video is continuous, about 10 minutes worth of webcam and audio are randomly missing, making the other 20 minutes very difficult to understand due to lack of continuity.

Session BA1 The pair takes over some code of unknown quality written by outsourced developers. Technically, they want to implement part of a cache. In particular, their logic should tell whether the requested data has changed since a given timestamp. In their session, they need to understand all existing code for that functionality (a few dozen lines of PHP code), make some additions, and encounter difficulties in specifying what exactly their cache should do. In the first minutes they also struggle with the workstation which is not theirs and not fully configured to their needs.

6.3. Company C

Company C develops a geographic information system desktop GUI application written in Java. The design of this software uses abstraction elaborately.

Over the course of one week, we recorded 6 sessions involving 8 developers in this company. See also Plonka's description of the data collection [4, pp. 64–67], and the (German) handout produced for company C in Appendix B. See Table 6 for general information on the developers and Table 19 for structured information on the recorded sessions.

Company	Sessions										Devs	Pairs	Auxiliary Data	
	Duration			Mode						Σ			Questionnaire	Reflection
	min	avg	max	PP	PPao	SbS	Mob	RPP	DPP				Pre	Post	Interview
A	2:22	2:22	2:22	1						1	2	1	✓	✓	–
B	1:21	1:37	1:51	4						4	2	1	✓	✓	–
C	1:12	1:28	2:10	6						6	8	6	✓	✓	(✓)
D	0:31	1:33	2:23	6						6	8	5	(✓)	✓	(✓)
E	1:17	1:47	2:46	7						7	8	6	✓	✓	(✓)
F	1:45	2:03	2:39	4						4	6	4	✓	✓	(✓)
J	0:42	1:53	5:27						9	9	2	1	–	(✓)	–
K	0:53	1:40	2:53	8						8	4	3	–	(✓)	(✓)
L	0:47	0:53	1:00					1	1	2	3	2	–	–	–
M	0:25	0:25	0:25	1						1	2	1	–	–	–
N	0:41	1:35	3:29			4	1			5	4	3	–	–	(✓)
O	0:47	1:13	1:44	2	2		4	2		10	5	6	–	–	(✓)
P	0:58	1:25	1:42	4						4	3	2	–	–	✓
Σ	0:25	1:35	5:27	43	2	4	5	3	10	67	57	41			

Table 3: Overview of the PP-ind repository of professional pair programming session recordings including the minimum, average, and maximum durations of each company's recordings as well as the recording count per collaboration mode (zeros are omitted for readability). The modes are explained in Section 4.1. Auxiliary data may be ✓–complete, (✓)–partial, or non-existing. Questionnaires are *complete* if they were handed out for all sessions then filled out by both partners individually; *complete* reflection interviews are all audio-recorded while *partial* means missing records or hand-written notes only.

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
A1	male	DE	10	3	2		4	3	1	1
A2	male	DE	7	5	2		7	5	1	1
								Pair constellations		**1**

Table 4: Overview of the A developers. See Section 4.2 for information on the data and its representation.

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
B1	male	DE	8	10	0	5	0	7	4	1
B2	male	DE	*n.a.*		0	6	*n.a.*		4	1
								Pair constellations		**1**

Table 5: Overview of the B developers. See Section 4.2 for information on the data and its representation.

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
C1	male	DE	4	2	2		*n.a.*		1	1
C2	male	DE	9	2 / 3	"ca. 6?"		8	5	2	2
C3	male	DE	6		0		5	9	1	1
C4	female	DE	10		11 / 12		6	2	3	3
C5	male	DE	"20+"		20		2	4	1	1
C6	female	DE	2	8	6	7	*n.a.*		1	1
C7	male	DE	10		12		6	4	2	2
C8	male	DE	0	11	3		1		1	1
								Pair constellations		**6**

Table 6: Overview of the C developers. See Section 4.2 for information on the data and its representation.

Session CA1 Developers C1 and C2 work on a new form on the system's GUI. C1 has started working on the form prior to the session; C2 is new to the task. In their session, they mostly deal with making their new GUI component in the form toggleable for which they reuse existing GUI logic.

Session CA2 Developers C2 and C5 work on a small functional extension and its main difficulty lies in understanding and properly applying the existing design abstractions. The task was started by C5 prior to the session. The work consists of some design-discussion and moving of classes to another package in the first half of the session and of implementation and testing of a new abstraction in the second half.

Session CA3 Developers C6 and C7 want to implement a new context menu entry which is only enabled under certain circumstances. They write test cases for the menu entry to be enabled and disabled, and refactor code along the way. This is a simple task, but their IDE freezes many times for over a minute each time which slows the pair down a lot. After 1:20 hours into their session, the pair took a break of four minutes.

Session CA4 Developers C4 and C7 implement a new feature to allow for selection of multiple graphical features while holding down the `CTRL` key. They have to adapt many interfaces in the event handling part of the software since their feature is the first to react to the `CTRL` key. They write tests, do refactorings, and discuss design a lot; quite some time is lost in the second half of the session due to a problem with the team's SVN (Subversion) server. The pair also uses two sets of keyboard and mouse fluently.

Session CA5 Developers C3 and C4 start implementing a new feature that allows users to cut existing geometries (such as points, lines, polygons) into parts by drawing arbitrary shapes across them. The pair uses two sets of keyboard and mouse fluently and generally have a productive and high-paced session.

Session CB1 Developers C4 and C8 work on a pet project of theirs before official office hours. It is written in Java, too, but has nothing to do with the domain of the C company.

6.4. Company D

Company D develops a large customer-relationship system that is based on Eclipse and comprises some 50 top-level modules written in Java.

During one week, we recorded 6 sessions involving 8 developers in this company. See also Plonka's description of the data collection [4, pp. 64–67]. See Table 7 for general information on the developers and Table 20 for structured information on the recorded sessions.

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
D1	female	DE	10		2		2	3	1	1
D2	male	DE	12		1	6	*n.a.*		3	2
D3	male	DE		3		3		3	1	1
D4	male	DE	1		"–"		0	0	2	2
D5	male	DE	9		1	8	1	8	1	1
D6	male	DE	11		*n.a.*		1	10	1	1
D7	male	–, DE	4		2		*n.a.*		1	1
D8	female	–, DE	3	2	3	8	*n.a.*		2	1
								Pair constellations		**5**

Table 7: Overview of the D developers. See Section 4.2 for information on the data and its representation.

Session DA1 *Note on Data:* No webcam was recorded.

Session DA2 Developer D4 is in his very first week at the company and has never pair programmed before; his colleague D3 has been with the company for three months. It is his first programming job for which he started learning Java.

In their session, they try to implement a new toolbar for one of the system's many modules. After some failed attemptes and two long discussions with additional developers (first D7, then D6), they perform a technically simple refactoring task which spans many different modules and lasts the rest of session. Throughout the session, D4 provides D3 with information on programming styles, technologies, and so on, whereas D3 is more knowledgeable about the code base and the organizational background.

The team had its daily stand-up meeting in the middle of the session which leads to a 15 minute break.

6.5. Company E

Company E develops a graphical desktop application for different logistics related tasks in the C++ programming language, but there are also parts written in C# and Java.

During one week, we recorded 7 sessions involving 8 developers in this company. See also Plonka's description of the data collection [4, pp. 64–67]. See Table 8 for general information on the developers and Table 21 for structured information on the recorded sessions.

Session EA1 Prior to the session, developer E2 already tried to debug a display error that leads to routes of ferries being displayed with an extra segment. In their PP session, he and colleague E1 go through the unfamiliar source code step by step with a debugger. They repeatedly set up a certain state, inspect variables, develop and test

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
E1	male	DE	0	5	0	5	*n.a.*		3	2
E2	male	DE	10	8	3 / 5		*n.a.*		3	2
E3	male	DE	23	0	0	0	*n.a.*		2	2
E4	male	DE	8	0	0	0	6		2	2
E5	male	DE	10		2		*n.a.*		1	1
E6	female	DE	13		0		*n.a.*		1	1
E7	male	DE	2			2	*n.a.*		1	1
E8	male	DE	2	5	2		2	7	1	1
								Pair constellations		**6**

Table 8: Overview of the E developers. See Section 4.2 for information on the data and its representation.

hypotheses, but they do not change any code. They end their session after 80 minutes because of a team meeting without having really circled in on the error.

Note on Data: There are several pauses in the video stream because the developers accidentally paused and resumed the recording underway by using the `F9` and `F10` keys in their debugger which the recording software also listened to.

6.6. Company F

Company F develops an email marketing tool written in Java.

On three consecutive days, we recorded 4 sessions involving 6 developers in this company. See also Plonka's description of the data collection [4, pp. 64–67]. See Table 9 for general information on the developers and Table 22 for structured information on the recorded sessions.

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
F1	male	DE	10		4		1	2	1	1
F2	male	DE	10		4		0	3	2	2
F3	male	DE	5		3		2	2	2	2
F4	male	DE	1	6	"rarely"		*n.a.*		1	1
F5	male	DE	15	"?"	5		7		1	1
F6	male	DE	1	4	1	4	1	4	1	1
								Pair constellations		**4**

Table 9: Overview of the F developers. See Section 4.2 for information on the data and its representation.

Note on Data: In all sessions, the pairs worked on a dual-screen setup but only one screen was recorded.

6.7. Context J

Company J is a service provider for public radio broadcasters. All recorded PP sessions involve the same pair of developers who do *distributed pair programming* because they work in different cities: J2 is directly employed by said company and J1 is a consultant. The pair synchronized their development environments with the Eclipse plugin "Saros",[1] which allows concurrent editing in all shared files for both developers.

Their 9 sessions were recorded on four days: Session **JA1** on one day, sessions **JA2** to **JA9** two weeks later on

1. Project homepage: https://www.saros-project.org

three consecutive days. See also Schenk's description of the data collection [13, pp. 137–139].

J1 and J2 have known each either for about one year, the first two months of which they shared an office. However, as a consultant, J1 first got in contact with the concrete application domain 8 months prior to the session recordings. See Table 10 for general information on the developers and Table 23 for structured information on the recorded sessions.

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
J1	male	DE	6	4	*n.a.*		0	8	9	1
J2	male	DE	1	6	*n.a.*		1	6	9	1
									Pair constellations	**1**

Table 10: Overview of the J developers. See Section 4.2 for information on the data and its representation.

Session JA1 Domain expert J2 had designed and implemented a plugin-based architecture in Java to monitor and download remote files from the servers of different radio stations about a year earlier. He invites experienced consultant J1 to a distributed pair programming session to review and clean-up the code together in to order ease the implementation of a new feature later on. In the session, they review but one class, try and fail to refactor it by extracting local methods, and ultimately decide to rewrite the whole system from scratch, which they do two weeks later in sessions **JA2** to **JA9**.

Session JA2 The pair starts with developing the module from scratch. In the first part of the session, J2 shows J1 a number of helper implementations he wrote in the meantime and J1 criticizes them. Afterwards, they discuss and collect requirements together in a plain text file.

6.8. Company K

Company K develops and operates a large web-portal for many estate-related services using different technologies which are separated through a microservice architecture.

We collected data on multiple occasions: Sessions **KA1** and **KA2** show an inter-team collaboration, sessions **KB1** and **KB2** show in-team work two months later, and **KC1** to **KC4** take place yet another six months later after the team had changed its technology stack. See Table 11 for general information on the developers and Table 24 for structured information on the recorded sessions.

Note on Naming Scheme: Originally, we considered all 8 K-sessions to be of the same context and numbered them accordingly "KA1" to "KA8". However, we later decided to split them into three different contexts due to shift in involved systems and even programming languages. Therefore, when we mention "session KA6" in [18], we actually refer to session **KC2**.

Session KA1 and KA2 Developers K1 and K2 come together to work out an API between their respective teams' subsystems: K1 is responsible for a mobile app for which K2 writes the endpoint with Java Spring web framework.[2] Before they can start with their actual task,

2. Project homepage: https://spring.io/

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
K1	male	DE	0	6	*n.a.*		0	6	2	1
K2	male	DE	*n.a.*		2	6	0	3	8	3
K3	male	DE	8		2 / 3		1		4	1
K4	male	DE	12		*n.a.*		1	3	2	1
									Pair constellations	**3**

Table 11: Overview of the K developers. See Section 4.2 for information on the data and its representation.

they first need to change the target URL of a single link which takes them more than 45 minutes and the help of two colleagues, because their development environment was not properly set up. Afterwards, K1 explains the data he needs with some dummy JSON file he prepared and K2 considers which internal microservices are able to provide which kind of data.

After a lunch break, the continue with a first implementation in session **KA2**. Overall, their pair work involves reading a lot of somewhat-known source code and existing API specifications, and generates a lot of fresh common ground between the two.

Sessions KB1 and KB2 These two sessions were recorded two months after session **KA1** and **KA2**. Developers K2 (who is now more experienced in the domain) and K3 (who is more of a database expert) amend their data model: First they introduce a new model class and discuss which fields to include. In the second half they write and debug a database migration to adapt the database schema.

Sessions KC1 and KC2 Yet another six months later, developers K2 and K3 work together in their now changed environment: The team switched its technology stack from Java to CoffeeScript.[3] The two are in the process of getting to know the jQuery JavaScript library[4] because they want to write an integration test of an auto-completion feature, for which they want to programmatically enter characters into an input field. In session **KC1**, they set up their test environment and discuss different test approaches. In session **KC2**, after a lunch break, they try out these approaches which does not work out as intented and they struggle with their debugger.

6.9. Context L

L1 is a freelance consultant who offers training sessions to individual developers who bring their own tasks to work on with him remotely.

We recorded two of these training sessions with two different clients L2 and L3. See Table 12 for general information on the developers and Table 25 for structured information on the recorded sessions.

Session LA1 Consultant L1 and developer L2 go through a tutorial on the Ruby language.[5] L1 is proficient in Ruby (but still learns some fundamental aspects) whereas L2 is fairly new to it.

3. Project homepage: https://coffeescript.org/
4. Project homepage: https://jquery.com/
5. Tutorial hompage: http://rubykoans.com/

ID	Gnd	Lang	Dev. Y	Dev. M	PP Y	PP M	Comp. Y	Comp. M	#S	#P
L1	male	EN	11		*n.a.*		2	7	2	2
L2	male	EN	14		*n.a.*		0	1	1	1
L3	male	EN	*n.a.*		*n.a.*		*n.a.*		1	1
								Pair constellations		**2**

Table 12: Overview of the L developers. See Section 4.2 for information on the data and its representation.

Session LB1 Consultant L1 and developer L3 play around with the impress.js library for browser-based slideshows.[6]

6.10. Company M

Company M develops software for multiple clients in the energy and logistics sector.

We recorded a single session with developers M1 and M2. See Table 13 for general information on the developers and Table 26 for structured information on the recorded sessions.

ID	Gnd	Lang	Dev. Y	Dev. M	PP Y	PP M	Comp. Y	Comp. M	#S	#P
M1	male	–, EN	8		*n.a.*		2	1	1	1
M2	male	–, EN	2	2	*n.a.*		2	2	1	1
								Pair constellations		**1**

Table 13: Overview of the M developers. See Section 4.2 for information on the data and its representation.

Session MA1 Developer M2 goes through a number of database tables and asks M1 many questions about their and their columns' purpose. Since M2 had prepared a list of `SQL SELECT` queries on the tables and columns in question their session is efficient and only lasts 25 minutes.

6.11. Company N

Company N develops and operates the web platform for a fashion retailer.

We recorded 5 sessions during the company's "on-boarding" process where a group of new-hires is introduced to the company and its technology stack. During these sessions, each developer worked on his or her own machine to set up the Docker-[7] and AWS-based[8] development environment. There is no source code involved, only a multitude of documentation which the developers try to work through in groups of two or three. See Table 14 for general information on the developers and Table 27 for structured information on the recorded sessions.

Session NA1 and NA3 Developers N1 and N2 sit next to each other and work through the documentation, but each of them configure their own machine. They take a break for about one hour between **NA1** and **NA3**.

Note on Data: While N1's audio, screen, and webcam are recorded, for his partner N2 only the audio is available.

6. Project homepage: https://impress.js.org
7. Project homepage: https://www.docker.com/
8. Amazon Web Services: https://aws.amazon.com/

ID	Gnd	Lang	Dev. Y	Dev. M	PP Y	PP M	Comp. Y	Comp. M	#S	#P
N1	male	DE	*n.a.*		*n.a.*		0	0	2	1
N2	male	DE	*n.a.*		*n.a.*		0	0	3	2
N3	female	DE	*n.a.*		*n.a.*		0	0	3	2
N4	male	DE	*n.a.*		*n.a.*		0	0	3	2
								Pair/group constellations		**3**

Table 14: Overview of the N developers. See Section 4.2 for information on the data and its representation.

Session NA2 and NA4 Developers N3 and N4 sit next to each other and work through the documentation, but each of them configure their own machine. They take a break for about one hour between **NA2** and **NA4**.

Note on Data: While N3's audio, screen, and webcam are recorded, for her partner N4 only the audio is available.

Session NA5 One week later, developers N2, N3, and N4 sit together at one table, each of them still trying to set up their respective development environments. They exchange ideas and insights for three and a half hours.

Note on Data: N4's webcam is not recorded, but all other channels are (i.e., 3× audio, 3× screencast, and 2× webcam).

6.12. Company O

Company O develops a web-based project planning tool using CoffeeScript in both frontend and backend.

During a four-week observation period we recorded 10 sessions on 4 recording days. The team employed all of normal pair programming, mob programming, side-by-side programming, and remote pair programming. They also spoke both German and English. See Table 15 for general information on the developers and Table 28 for structured information on the recorded sessions.

ID	Gnd	Lang	Dev. Y	Dev. M	PP Y	PP M	Comp. Y	Comp. M	#S	#P
O1	male	DE, EN	10	8	*n.a.*		1		7	4
O2	male	DE, EN	*n.a.*		*n.a.*		0	5	2	1
O3	female	–, EN, DE	1	1	*n.a.*		0	1	9	5
O4	male	DE, EN	*n.a.*		*n.a.*		0	5	9	5
O6	male	DE, EN	*n.a.*		*n.a.*		*n.a.*		1	1
								Pair/group constellations		**6**

Table 15: Overview of the O developers. See Section 4.2 for information on the data and its representation.

Sessions OA1 and OA2 Developers O3 and O4 are tasked with writing a test case for some new functionality. Even though they get some help from a colleague along the way, they do not make progress in sessions **OA1** and **OA2** (separated by a lunch break). There are multiple reasons for this: They neither know that part of the production code nor the underlying technology (React and Redux[9]) nor their development environment so they resort to "console.log"-debugging for which they have to rebuild the software in three-minute cycles. The pair

9. Project homepages: https://reactjs.org/ and https://redux.js.org/

speaks English throughout the session, which is neither developer's first language.

Note on Data: Session **OA1** started as plain PP with only one screen. However, O4 started using his laptop along the way (and continued in **OA2**), but his screen is not recorded. Additionally, there is not webcam recording for session **OA1**.

Sessions OA3, OA4, and OA5 Four developers of the team work on a bug that causes some rendering issues on drag-and-drop actions in a calender view. Between **OA3** and **OA4** there is a 10 minute break; between **OA4** and **OA5**, there is a 15 minute break, during which developers O2 and O4 left the group, leaving just the pair O1/O3 who amend test cases, refactor the production code, and eventually fix the bug. Along the way, O1 explains general software development principles to O3.

Sessions OA6 and OA7 Developers O3 and O4 try to understand a performance issue in their web application for about an hour. Twenty minutes later, O4 continues this task with O1. Both sessions are done via a Skype call with audio and screensharing but no webcam.

Note on Data: No webcam was recorded.

Sessions OA8, OA9 and OA10 The three developers O1, O3, and O4 work on a bug in three sessions. At first, they investigate both production and test code to understand the reason for a newly failing test case that is unrelated to the bug: They changed some implementation but did not adapt the mock objects used in the tests accordingly. They eventually adapt the mock objects and write new test cases to reproduce the bug. Developer O1 leaves session **OA8** after 17 minutes, but is again part of the group in sessions **OA9** and **OA10**.

6.13. Company P

Company P develops and operates the web platform for a car part retailer. The website is consists of multiple large PHP apps.

During a one-week stay, we recorded 4 sessions with 3 developers. See Table 16 for general information on the developers and Table 29 for structured information on the recorded sessions.

ID	Gnd	Lang	Dev.		PP		Comp.		#S	#P
			Y	M	Y	M	Y	M		
P1	male	DE	5		2		2		4	2
P2	male	DE	6		3	6	3	6	2	1
P3	male	DE	5		2		2		2	1
								Pair/group constellations		**2**

Table 16: Overview of the P developers. See Section 4.2 for information on the data and its representation.

Sessions PA1 and PA2 In session **PA1**, developers P1 and P2 review a database migration written by P1 and later discuss the requirements that led to the database schema change in the first place. They continue after their lunch break with session **PA2** where they test and debug the migration and end up changing test cases to remove embedded false assumptions.

Sessions PA3 and PA4 Developers P2 (who is more proficient in the backend) and P3 (more frontend experience) continue the implementation of a new API endpoint which P3 already started. In **PA3**, P3 shows his existing implementation for which they write tests; P2 explains some backend-related software development best practices. On the next day, in session **PA4**, they continue with implementing the database access which causes them problems because of some idiosyncrasy of their object-relational (OR) mapper. There is a 13-minute break in session **PA4** due to a spontaneous team meeting.

7. Discussion

The data collection procedure described in Section 3 has a number of properties which affect the data quality in terms of the kind of data which can be collected this way, but also affecting the developers' behavior.

7.1. Limitation of Scope

Due to the method's design, a number of systematic gaps can be expected. That is, there probably exists industrially relevant pair programming behavior which is *not* captured.

- **Not all companies:** Due to our naturalistic approach, we did not ask developers to pair program, so we did not target companies with little or no pair programming usage.
- **Not all developers:** All recordings are voluntary and some developers may not want to be recorded. In company P, for example, one team member was generally inclined to working in pairs, but did not want to be part of the data collection, despite an emphatic recommendation by a colleague who just had his first reflective interview.
- **No short sessions:** The majority of the sessions in the repository is one hour or longer, with the shortest one being about 25 minutes long. In discussions with practitioners, however, some reported of regular session lengths of 10 or 15 minutes. The recording infrastructure is designed for recording individual sessions, not, say, everything that happens on a work station over the course of a day. Since the recording setup poses an overhead to the normal work flow, *ad hoc* pairings are difficult to record. Furthermore, pairs that had already gone through organizing and setting up a recording session then possibly work longer than they otherwise would have.
- **No tense situations:** The mere presence of a researcher on site may be regarded as a distraction. In both companies O and P there were multiple months between the first discussion and the start of the main data collection, and in both cases it was due to the Scrum Masters wanting to postpone the research activity until a turbulent phase in the respective projects was over. A second data collection phase in company O did not happen because of immense time pressure for the software developers—even though both developers and Scrum Masters were very happy with the insights from the first round. It is not clear whether any pair programming was done in these stressful phases.

There are other limitations of the data, which are not strictly due to design, but due to practicalities of getting in contact with a company and traveling.

- **Western Cultural Background:** All companies are based in Germany with the exception of company M which is in Oslo, Norway. The L-sessions were individual training sessions and had no organizational background (see Section 6.9). Most developers are native German speakers. The L-developers are the only native English speakers; the M- and O-developers use English as their work language.

7.2. Effects of Recording Infrastructure

In companies A to D, the developers' IDEs were also equipped with a plugin to collect technical information on their current activities, focus, etc. (see [9, pp. 85 & 461–479]). This led to some artifacts in the programming sessions, e.g., in **CA2** where the developers spent 1:20 minutes trying (and failing) to look up an ID from the issue tracker on the (remote) development computer before tabbing out of the RDP session to find the necessary information within five seconds on the (local) recording computer which they first avoided to ensure continuous data collection. In session **CA3**, the IDE was repeatedly unresponsive for 1:20 minutes at a time totaling about one third (!) of the whole session, for which the developer had no explanation and which may be due to a defect in the data collection plugin.

There were several instances of developers not working on their own machine. In sessions **CA2** and **BA1** the developers spent some time to get comfortable with their IDE as some options are not set as they were used to. Session **DA2** starts with several minutes of the pair waiting for an SVN update to complete since the workspace had not been used for a while.

The recording infrastructure was not always fully compatible with the local circumstances and the developers' habits. In session **EA1**, the same keyboard shortcut was used for stepwise debugging as for Camtasia to pause recording. Not only did this lead to some gaps in the screencast, but appears to have also slightly confused the developers (as the continuous audio recording reveals). All pairs in company F appear to have used two monitors, but the recording setup at the time was not able to capture both, so the screencast is missing one half.

Although wireless microphones and webcam were supposed to not bother the developers, they occasionally fiddled with it, e.g., before leaving the desk for a minute and again upon their return. One pair knocked over the webcam from its tripod; another pair took a break together to get some candies and wandered beyond the wireless transmitter radius while still talking about their task.

Reports on how the subjects felt regarding the data collection are available from the C-developers only, some of which say they felt being watched while others claim to have forgotten the camera after five minutes.

7.3. Effects of Pre-Existing Notions

We had pre-existing notions of what the social reality of industrial software development looks like which were deeply embedded in the data collection process.

Not Recording All Aspects The first such notion is *pair programming* itself. Our research started with the textbook definition of 'two developers jointly working on one computer'. Neither of these quantities is fixed in everyday industrial development, but they *are* in the session recording infrastructure: Screencast software, microphones, and camera angle are all set up for *two* developers on *one* computer. However, developers may suddenly open their own laptop or interact with others developers who are out of reach of the microphones and/or beyond the camera angle (as was noted by developer D4 in session **DA2** after a third developer joined the pair: *"We need more microphones!"*).

Another impact from the research interest on the way data was collected can be seen in the sessions recorded by Plonka, who was initially interested in the *driver/navigator* metaphor. In order to easily see who is control of keyboard and mouse, the camera angle of sessions in companies C, D, E, and F centers on the developers' hands, but occasionally cuts off their faces. Sessions from the companies focus on the developers' faces instead.

In both cases, possibly relevant aspects of the PP process are not recorded making the reconstruction more difficult. In the first, there is technical information missing, in the second, the developers faces and viewing direction makes interpreting their behavior more difficult at times.

Affecting Developer Behavior The second, related notion is that of a *session* pertaining to a *task* as a meaningful unit the software developers' workday. Some companies form pairs independent of concrete tasks for multiple days on end during which the pair members behave as *one*, taking coffee breaks together without really starting or ending a "session". The data collection procedure described above is session-centric. Questionnaires before and/or after the recording frame the session in two senses. First, they set a ceremonial start and end: In the beginning of sessions **CA2** and **EA1**, the developers filling out the questionnaire was accidentally recorded—it took them more than nine minutes, which might be an unnatural break. Second, the pre-session questionnaire asked the developers to think about the work time ahead. In particular, the questionnaire asked for task classification and description, a characterization of the expected difficulties, and an estimated time to completion (see Fig. 2). Although this may yield valuable context information for the researcher, it may impose an unnatural focus on the developers by making them think about aspects they would not have thought about had it not been for the recording.

Another effect of session-centrism can be observed in multiple session recordings in various companies: Pair members are occasionally interrupted by their colleagues with technical or organizational concerns. A common reaction of the pairs in the recordings is to send away the interrupter unsatisfied (as seen in session **AA1** at 1:47:05) or to point out that they just walked into a recording (as seen in session **KA1** at 0:04:30), as if the pair programmers wanted to protect the integrity of

the data collection.[10] The recordings in company E are peculiar in another way, which also possibly indicates an intention of the developers to protect the data collection: The work station for the session recordings was set up in a meeting room, such that the pairs worked completely secluded from the rest of their team. Apparently this was the company's decision, but it seems unlikely that this reflects the team's normal pair programming practice.

Such effects are more pronounced in the recordings that were done under a "Understand PP" headline (see Table 1). For instance, the pairings in companies C to F do not appear to be holding up to the naturalistic ideal as only *one* of the overall 21 pairs was recorded twice.[11] In the spirit of a "workshop", the developers could write their names on a list with a morning and an afternoon slot if they wanted to be recorded (see also Appendix B).

In company P, the team at one point discussed how the next pair should be formed based on what we at the time understood as organizational constraints. However, in the next reflective interview, the developers revealed to us that they intended to give another, previously not-recorded colleague the chance to also benefit from the feedback we could provide. Again, the employed data collection method did interfere with a 'natural' formation of pairs.

7.4. Summary of Data Quality

Notwithstanding the above limitations of the data collection—some a direct result of the way the involved researchers collected data (limited breadth of data collection which complicates interpretation, also: framing), others resulting from reactions of the subjects to the researchers' presence (protecting integrity, longer sessions)—the session repository still comprises diverse, realistic, detailed data. At the time of writing, it contains 67 recordings from 13 different companies featuring 57 different professional software developers who worked together in 41 different constellations of (mostly) two members.

In all of these sessions, the developers worked on their normal tasks for as long as they wanted, and in most cases, the developers also freely chose who to work with and when to start. The exception here are the 23 sessions from companies C to E, where the developers had to sign up for either a morning of an afternoon slot, and were possibly inclined to work with partners they would normally not pair with in the prospect of learning something in the reflective interview. It can also not be ruled out that the developers in all companies generally felt compelled to work in pairs more often than normal for the same reason.

All of the above concerns may affect the *frequency* of phenomena (such as more or less conflict situations, more or less easy tasks, or more or less fatigue due to longer sessions), but none of these appear to produce to entirely artificial behavior. Depending on the make up of concrete studies (e.g., qualitative or quantitative approach), the above considerations need to be kept in mind.

10. Colleagues could only 'walk in' on a session recording because (a) in company K the webcam was not mounted on a highly visible tripod but on the monitor (unlike in companies C to F, see discussion of camera angles above), and (b) the recording was done remotely with no researcher on-site to notice. This inadvertent 'covert recording' of uninitiated colleagues is an ethical concern that only occurred to us while writing this report. In this sense, informing the colleague that he is now on record can also be interpreted as *protecting their privacy* more than *protecting the data collection*.

11. The pair E1/E2 worked together in **EA1** and **EA6**. Sessions **DA5** and **DA6** also feature the same pair (D2/D8), but were recorded in one sitting.

8. Usage in Publications

See Figure 4 for an overview of all publications from our research group that build on the data of the PP-ind repository described in this document.

References

[1] Erik Arisholm, Hans Gallis, Tore Dybå, and 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.

[2] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 1999.

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

[4] Laura Plonka. *Unpacking collaboration in pair programming in industrial settings*. PhD thesis, Open University, 2012.

[5] Laura Plonka, Judith Segal, Helen Sharp, and Janet van der Linden. Collaboration in pair programming: Driving and switching. In Alberto Sillitti, Orit Hazzan, Emily Bache, and Xavier Albaladejo, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 77 of *Lecture Notes in Business Information Processing*, pages 43–59. Springer, 2011.

[6] Laura Plonka, Judith Segal, Helen Sharp, and Janet van der Linden. Investigating equity of participation in pair programming. In *Proc. Agile India 2012*, 2012.

[7] Laura Plonka, Helen Sharp, and Janet van der Linden. Disengagement in pair programming: does it matter? In *Proc. 2012 Intl. Conf. on Software Engineering*, pages 496–506. IEEE Press, 2012.

[8] Laura Plonka, Helen Sharp, Janet van der Linden, and Yvonne Dittrich. Knowledge transfer in pair programming: An in-depth analysis. *Int'l. J. of Human-Computer Studies*, 73:66–78, 2015.

[9] Stephan Salinger. *Ein Rahmenwerk für die qualitative Analyse der Paarprogrammierung*. PhD thesis, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 2013.

[10] Stephan Salinger, Laura Plonka, and 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(1):9–25, 2008.

[11] Stephan Salinger and Lutz Prechelt. *Understanding Pair Programming: The Base Layer*. BoD, Norderstedt, Germany, 2013.

[12] Stephan Salinger, Franz Zieris, and Lutz Prechelt. Liberating pair programming research from the oppressive driver/observer regime. In *Proc. 2013 Int'l. Conf. on Software Engineering*, ICSE '13, pages 1201–1204, Piscataway, NJ, USA, 2013. IEEE Press.

[13] Julia Schenk. *Industrially Usuable Distributed Pair Programming*. PhD thesis, 2018.

[14] Julia Schenk, Lutz Prechelt, and Stephan Salinger. Distributed-pair Programming can work well and is not just Distributed Pair-programming. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 74–83, New York, NY, USA, 2014. ACM.

[15] David Socha, Robin Adams, Kelly Franznick, Wolff-Michael Roth, Kevin Sullivan, Josh Tenenberg, and Skip Walter. Wide-field ethnography. In *Proc. 38th Int'l. Conf. on Software Engineering Companion - ICSE '16*. ACM Press, 2016.

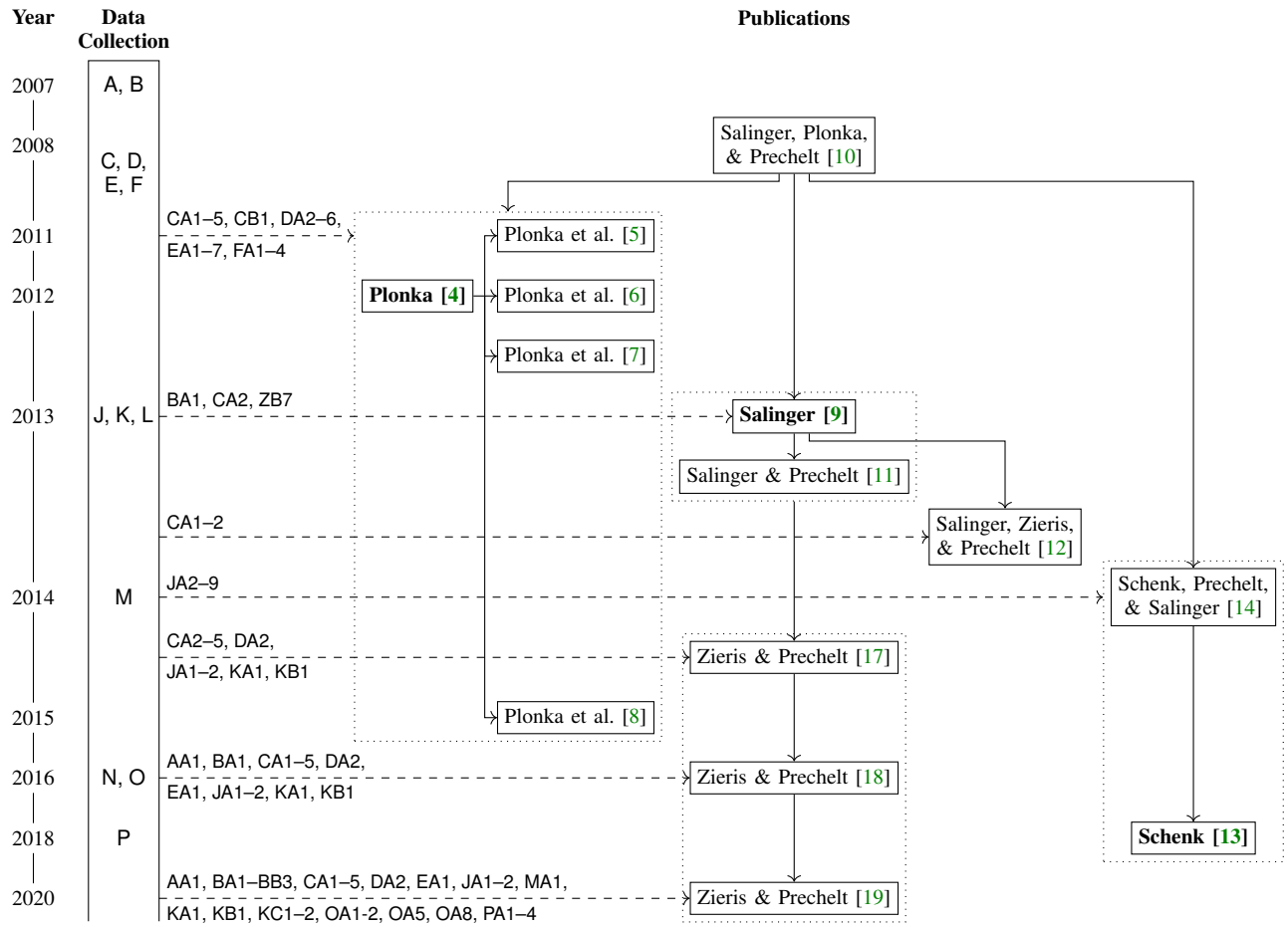


Figure 4: Timeline of data collection and scientific publications originating in our research group. Arrows between publications ⟶ indicate reuse of ideas or building on results; arrows from data collection - -› indicate which PP sessions were analyzed. PhD theses are set **bold**.

ID	Mode	Developer A1	Developer A2	Start	Dur.	SL	PL	Pre	Post
AA1	PP	X	X	2007-01-26 13:43	2:22	DE	Java, Objective-C, SQL, HTML, Tcl	2 1	2 1

Table 17: Overview of the A sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer B1	Developer B2	Start	Dur.	SL	PL	Pre	Post
BB1	PP	X	X	2007-04-27 13:25	1:21	DE	PHP, HTML	2 2	*n.a.*
BB2	PP	X	X	2007-04-27 16:51	1:51	DE	PHP, HTML, SQL, JavaScript, CSS	*n.a.*	*n.a.*
BB3	PP	X	X	2007-04-27 18:58	1:32	DE	PHP, HTML, JavaScript	*n.a.*	*n.a.*
BA1	PP	X	X	2007-09-14 13:38	1:47	DE	PHP	1- 2	2 2

Table 18: Overview of the B sessions. See Section 4.3 for information on the data and its representation.

[16] David Socha and Kevin Sutanto. The "pair" as a problematic unit of analysis for pair programming. In *Proc. Eighth Int'l. Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '15, pages 64–70, Piscataway, NJ, USA, 2015. IEEE Press.

[17] Franz Zieris and Lutz Prechelt. On knowledge transfer skill in pair programming. In *Proc. 8th ACM/IEEE Int'l. Symposium on Empirical Software Engineering and Measurement*, ESEM '14, pages 11:1–11:10, New York, NY, USA, 2014. ACM.

[18] Franz Zieris and Lutz Prechelt. Observations on knowledge transfer of professional software developers during pair programming. In *Proc. 38th Int'l. Conf. on Software Engineering Companion*, ICSE '16, pages 242–250, New York, NY, USA, 2016. ACM.

[19] Franz Zieris and Lutz Prechelt. Explaining pair programming session dynamics from knowledge gaps. In *Proc. 42nd Int'l. Conf. on Software Engineering (ICSE '20)*, 2020.

[20] Franz Zieris and Stephan Salinger. Doing scrum rather than being agile: A case study on actual nearshoring practices. In *Proc. 8th IEEE Int'l. Conf. on Global Software Engineering*, 2013.

ID	Mode	Developer								Start	Dur.	SL	PL	Pre	Post
		C1	C2	C3	C4	C5	C6	C7	C8						
CA1	PP	X	X							2008-05-05 13:27	1:18	DE	Java	3 3	2 2
CB1	PP				X				X	2008-05-06 07:54	1:12	DE	Java	2 1	2 2
CA2	PP		X			X				2008-05-07 11:46	1:14	DE	Java	3 4	3 1
CA3	PP						X	X		2008-05-07 15:34	2:10	DE	Java	2 2	1 1
CA4	PP				X			X		2008-05-08 10:25	1:34	DE	Java	1 1	2 2
CA5	PP			X	X					2008-05-09 10:32	1:23	DE	Java	4 4	2 3+

Table 19: Overview of the C sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer								Start	Dur.	SL	PL	Pre	Post	
		D1	D2	D3	D4	D5	D6	D7	D8							
DA1	PP	X	X							2008-10-06 14:07	2:21	DE	Java, XML	"3–4"	1	1
DA2	PP			X	X					2008-10-08 10:13	2:23	DE	Java	"first time"	2	2.5
DA3	PP				X	X				2008-10-08 14:01	1:05	DE	XML (Spring)	6	2.5	5
DA4	PP						X	X		2008-10-08 16:22	2:00	DE	Java, XML	3	2	"not done"
DA5	PP		X						X	2008-10-09 10:27	0:31	DE	Java	1	*n.a.*	
DA6	PP		X						X	2008-10-09 13:01	0:58	DE	Java	*n.a.*	"1 / 4"	1

Table 20: Overview of the D sessions. See Section 4.3 for information on the data and its representation. All pairs filled out the pre-session questionnaire together. In session **DA2**, the pair talks with two other developer (5 minutes with D7, then 12 minutes with D6). In session **DA6**, the pair worked on two somewhat separate tasks which D2 rated separately in the post-session questionnaire.

ID	Mode	Developer								Start	Dur.	SL	PL	Pre		Post	
		E1	E2	E3	E4	E5	E6	E7	E8								
EA1	PP	X	X							2008-10-27 11:29	1:17	DE	C++	"2–3"	3	2	3
EA2	PP			X	X					2008-10-27 13:18	2:46	DE	XML (Maven)	"first time"	"–"	1	1
EA3	PP	X				X				2008-10-28 10:43	1:25	DE	C++	3	3	3	"3–2"
EA4	PP				X		X			2008-10-29 09:37	1:52	DE	C#	"–"	"0"	2	2
EA5	PP			X				X		2008-10-29 13:09	1:41	DE	C++, Java	"none"	"5–6"	2	2
EA6	PP	X	X							2008-10-30 10:05	1:40	DE	C++	"2–3"	3	2	2
EA7	PP		X						X	2008-10-31 09:25	1:50	DE	C++	2-	2-	2	2+

Table 21: Overview of the E sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer						Start	Dur.	SL	PL	Pre	Post
		F1	F2	F3	F4	F5	F6						
FA1	PP	X	X					2008-11-11 15:06	1:45	DE	Java, XML, SQL	3 3	2 2
FA2	PP		X	X				2008-11-12 11:07	1:59	DE	Java, XML, SQL	3 3	1 2
FA3	PP			X	X			2008-11-13 11:06	2:39	DE	Java	2 3	2 1
FA4	PP					X	X	2008-11-13 15:18	1:52	DE	Java	2 2	3 1

Table 22: Overview of the F sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer		Start	Dur.	SL	PL
		J1	J2				
JA1	DPP	X	X	2013-01-31 14:05	1:07	DE	Java
JA2	DPP	X	X	2013-02-13 10:51	1:15	DE	Java
JA3	DPP	X	X	2013-02-13 13:17	1:53	DE	Java
JA4	DPP	X	X	2013-02-13 15:26	2:01	DE	Java
JA5	DPP	X	X	2013-02-14 10:33	1:36	DE	Java
JA6	DPP	X	X	2013-02-14 13:13	0:42	DE	Java
JA7	DPP	X	X	2013-02-14 14:54	1:56	DE	Java
JA8	DPP	X	X	2013-02-15 10:54	1:06	DE	Java
JA9	DPP	X	X	2013-02-15 12:59	5:27	DE	Java

Table 23: Overview of the J sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer				Start	Dur.	SL	PL
		K1	K2	K3	K4				
KA1	PP	X	X			2013-03-14 10:37	2:00	DE	Java
KA2	PP	X	X			2013-03-14 13:15	2:53	DE	Java
KB1	PP		X	X		2013-05-02 13:45	0:53	DE	Java, SQL
KB2	PP		X	X		2013-05-02 15:26	1:36	DE	Java, SQL
KC1	PP		X	X		2013-10-29 11:24	0:59	DE	CoffeeScript
KC2	PP		X	X		2013-10-29 12:59	2:01	DE	CoffeeScript
KC3	PP		X		X	2013-11-29 11:00	0:53	DE	CoffeeScript
KC4	PP		X		X	2013-11-29 11:44	2:10	DE	CoffeeScript

Table 24: Overview of the K sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer			Start	Dur.	SL	PL
		L1	L2	L3				
LA1	RPP	X	X		2014-02-27 20:41	1:00	EN	Ruby
LB1	DPP	X		X	2014-03-06 16:11	0:47	EN	HTML, CSS

Table 25: Overview of the L sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer		Start	Dur.	SL	PL
		M1	M2				
MA1	PP	X	X	2014-10-16 11:42	0:25	EN	SQL

Table 26: Overview of the M sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer				Start	Dur.	SL	PL
		N1	N2	N3	N4				
NA1	SbS	X	X			2016-01-12 13:30	0:47	DE	–
NA2	SbS			X	X	2016-01-12 13:37	0:41	DE	–
NA3	SbS	X	X			2016-01-12 15:06	0:57	DE	–
NA4	SbS			X	X	2016-01-12 15:12	2:04	DE	–
NA5	Mob		X	X	X	2016-01-18 12:46	3:29	DE	–

Table 27: Overview of the N sessions. See Section 4.3 for information on the data and its representation.

ID	Mode	Developer					Start	Dur.	SL	PL
		O1	O2	O3	O4	O6				
OA1	PPao			X	X		2016-06-01 10:51	1:24	EN	CoffeeScript
OA2	PPao			X	X	X	2016-06-01 13:27	1:32	EN	CoffeeScript
OA3	Mob	X	X	X	X		2016-06-08 14:55	0:59	DE, EN	CoffeeScript
OA4	Mob	X	X	X	X		2016-06-08 16:05	0:50	DE, EN	CoffeeScript
OA5	PP	X		X			2016-06-08 17:11	1:09	EN	CoffeeScript
OA6	RPP			X	X		2016-06-09 14:00	0:56	EN	CoffeeScript
OA7	RPP	X			X		2016-06-09 15:17	1:39	DE	CoffeeScript
OA8	PP	X		X	X		2016-06-15 13:47	1:16	EN	CoffeeScript
OA9	Mob	X		X	X		2016-06-15 15:16	0:47	EN	CoffeeScript
OA10	Mob	X		X	X		2016-06-15 16:50	1:44	EN	CoffeeScript

Table 28: Overview of the O sessions. See Section 4.3 for information on the data and its representation. Session **OA2** started as a PP (with Active Observer) session, and developer O6 joined for a period of 14 minutes. Session **OA8** started as a Mob Programming session, but O1 had to leave for a meeting after 17 minutes.

ID	Mode	Developer			Start	Dur.	SL	PL
		P1	P2	P3				
PA1	PP	X	X		2018-06-05 11:24	0:58	DE	PHP
PA2	PP	X	X		2018-06-05 13:35	1:30	DE	PHP
PA3	PP	X		X	2018-06-06 12:23	1:31	DE	PHP
PA4	PP	X		X	2018-06-07 11:09	1:42	DE	PHP

Table 29: Overview of the P sessions. See Section 4.3 for information on the data and its representation.

Appendix A.
Recording Technicalities

We used three different generations of recording setups, the first of which was developed by Laura Plonka and Stephan Salinger, the other two by Julia Schenk and Franz Zieris.

Generation 1

Two computers are used in this setup: The development environment of the developers runs on one machine whose output is mirrored to another computer where the screen content is recorded with TechSmith Camtasia.[12] On Linux machines (company A), developers would work locally while the screen content is transferred to a Windows machine via VNC (Virtual Network Computing) to be recorded remotely (see Fig. 5b). On Windows machines (companies C, D, E, F), the developers would sit on the recording machine and use RDP (Remote Desktop Protocol) to log on the development machine, such that the recording device only needs to run Camtasia and the RDP client (see Fig. 5c). In some cases (company B), the same machine would be used for both development and recording (see Fig. 5a) thus limiting the available resources available in the developers' environment.

For the audio channel, there were also two solutions. The easy one was to use the webcam microphone (companies A and B, Figs. 5a and 5b), which was not able to pick up the developers' speech independently making it difficult to understand them when both speak at the same time. For higher audio quality, two wireless microphones were used in later recordings (companies C to F, Fig. 5c). Using an external USB soundcard, the two mono channels were hard-panned (100% left and 100% right) and mixed to one stereo channel.[13]

Generation 2

For the recordings in companies J and K, we used the web conferencing tool Adobe Connect[14] which relies on a Flash-based browser plugin to share one's screen content, webcam, and microphone.

To record the distributed pair programming sessions of company J, two Adobe connect meetings were started such that both developers each shared their respective screen, webcam, and headset microphone (which they had to use anyway for their DPP session). On the receiving end, the researcher stacks the two screen outputs on top of each other on a vertically rotated monitor and records the whole configuration (see Fig. 5d). Schenk organized the setup and the recording of the J-sessions; see [13, Sec. 3.2.2 & 3.2.3] for more details.

In companies K and M a similar setup was used: The developers put the headbands of USB headsets in the nape of their necks such that the cushions do not cover their ears and the microphone can pick up their speech. Again, two Adobe Connect meetings were necessary, this time to transmit both microphone signals. In company K, the pairs' dual-screen setup could also be captured this way (see Fig. 5e).

Compared to the first generation, the advantages of this infrastructure were (a) that the session could be watched live by the researcher while it was still recording thus making the next step (quick analysis) a bit faster and (b) the ease of the setup allowed that the recording could be done completely remotely as the developers had all necessary equipment on-site (modern web-browser, webcam, and headsets). Disadvantages were that (c) this setup only worked for Windows machines and that support for Flash (necessary for running Adobe Connect) was already declining at the time, and (d) fluctuation in network latency for the two audio channels meant that, for co-located pair programming in companies K and M, there was a notable randomly changing offset between the two signals. Whenever both microphones picked up the sound of one developer, e.g., because they looked at each other, an annoying robotic echo could be the result, which makes listening to the record rather unpleasant.

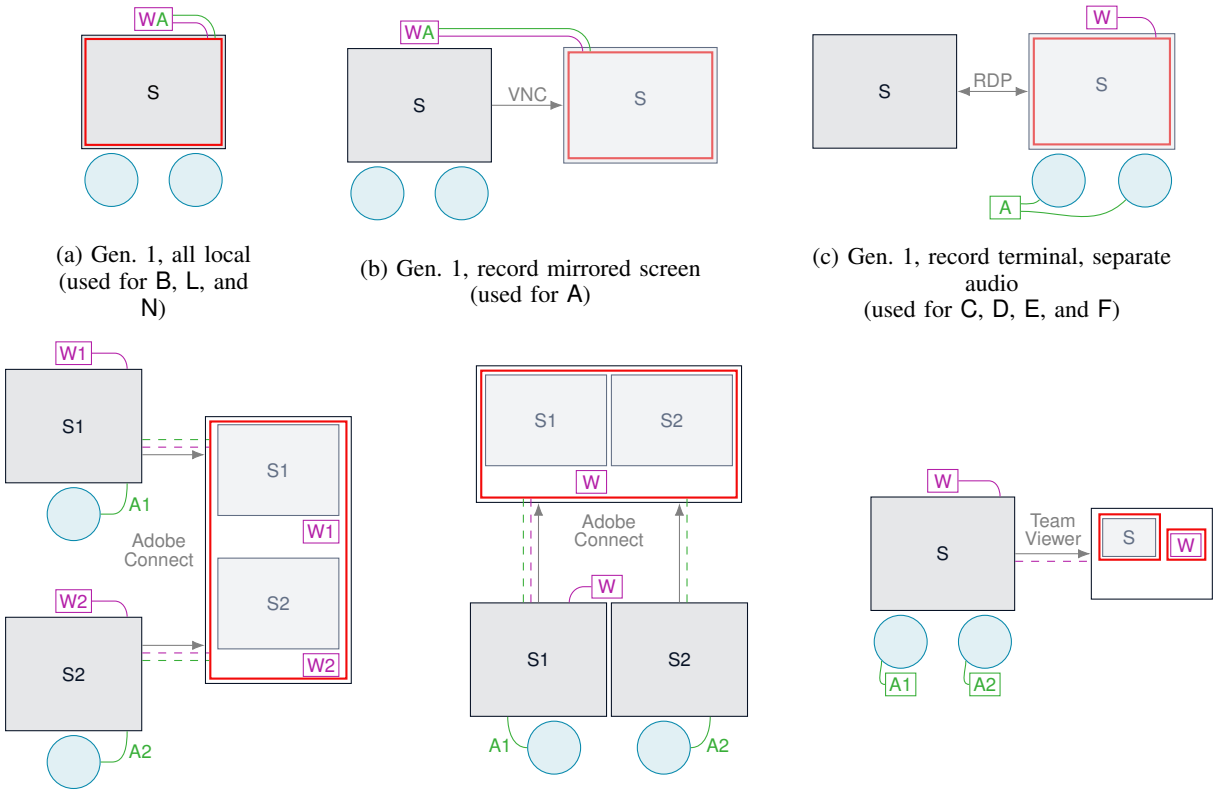
Generation 3

In order to enable recordings of pair programming sessions independent of the developers' operating systems, we looked for other solutions than Adobe connect. The most versatile ones turned out to be Skype[15] and TeamViewer,[16] the latter of which was able to transmit and record pixel-accurate screen contents so our research group purchased a commercial license. TeamViewer's recording feature allows to losslessly record the contents of a, say, Full HD display even if the recording device has a much smaller resolution available such as a window on a Laptop screen (see Fig. 5f). In order to process TeamViewer data further, however, the recording first needed to be rendered as a pixel video, which at times took much longer than the recording itself.

There were two options for recording the webcam. For relatively small distances between the development and the recording computers, the webcam could be connected directly to the recording machine using a really long USB cable in order to not disturb the developers. For setups more than one room apart, the webcam feed was transmitted in the TeamViewer session, which posed additional problems: Even though the webcam video is included in the pixel-video export, it has a fixed position and small resolution (ca. 160x120 pixels, next to the screen cast, in the upper right corner), and is also often notably out of sync with the screencast. To compensate this, we used Camtasia to screencapture the webcam feed separately while the session is running at a resolution of about 460x340 (see schematic in Fig. 5f, and annotated photograph from recording on-site in Fig. 6).

To avoid the robotic echo of network-transmitted dual audio, we record the audio of co-located sessions locally without any network in between. In company O, we used the same wireless microphones as before and fed them into one dictaphone; in company P we used two independent

12. Product homepage: https://www.techsmith.com/video-editor.html
13. Model of the microphones: Audio-Technica W-701/L; Soundcard: Tascam US-122L
14. Product homepage: https://www.adobe.com/products/adobeconnect.html
15. Product homepage: https://www.skype.com/
16. Product homepage: https://www.teamviewer.com/



(a) Gen. 1, all local (used for B, L, and N)

(b) Gen. 1, record mirrored screen (used for A)

(c) Gen. 1, record terminal, separate audio (used for C, D, E, and F)

(d) Gen. 2, record distributed PP (used for J)

(e) Gen. 2, record remotely (used for K and M)

(f) Gen. 3, record remotely, separate audio and webcam (used for O and P)

Figure 5: Different recording setups. Developers ◯ sit in front of a screen S with a webcam W and some means to record audio A. Screen contents are transferred → to another machine (together with audio - - - and/or webcam signal - - -), where they are recorded ▭ with a screen capture tool.

Figure 6: Concurrent recording of screencast and webcam as shown in Fig. 5f. This is an annotated photograph taken during the recording of session **PA2** in a conference room at company P. The developers P1 and P2 sit two rooms further down the hall. We used an external monitor only to follow the session live; it would not have been strictly necessary because TeamViewer will record the screencast in full resolution regardless of its current display size.

dictaphones each wired to a simple lapel microphone to record the developers individually.[17]

Synchronizing the Data Channels

Depending on the setup, a recording session would yield a number of files with a number of data streams. In the simplest form (as for companies A and B) the result would be a single Camtasia file containing the screencast stream and the webcam stream (which also includes the audio). Such a file could be opened directly in Camtasia Studio and exported to a self-contained video file such as an AVI or MP4 video file.

In companies C through F—where audio was recorded externally—there were two Camtasia files: One comprising the screen content and webcam video just as in the simplest setup; the other containing the externally recorded dual-mono-mixed-to-stereo audio signal. To ease synchronizing the separate audio and video channels, the developers in these recordings were asked to clap in front of the camera. Ultimately, there were two 'timelines' that needed to be aligned.

With the second generation recording infrastructure using Adobe Connect web conferences, no such alignment was strictly necessary as all individual streams (two screens, one or two webcams, two audio channels) were already synchronized the moment they were captured off the researcher's screen. However, some audio post-processing was still necessary as Adobe Connect center-panned both audio channels during the meeting such that they were mixed together in the Camtasia recording. Luckily, Adobe Connect allowed to record the sessions on the conference server and provided a large ZIP file containing all individual streams, including the audio track as an MP3 file. Using the mixed audio from the Camtasia recording as a reference, we could align the two separate audio recordings and hard-pan them to ultimately get a self-contained video after aligning these three timelines.

In the third recording generation, which relies on TeamViewer, a total of *four* timelines need to be synchronized: The screencast; two independent dictaphone recordings which, even though they come from the same model with identical settings, differ inexplicably in recording speed by several seconds per hour; and the webcam feed. To synchronize these, we first stretched one of the audio recordings (without changing the pitch) to match the other, then looked for isolated code changes in the screencast and listen for audible keystrokes, then noticed that the either the screencast or audio needed to be stretched again as they, too, drifted further and further apart towards the end of the session, and finally did the same again for the webcam video (looking for bilabial plosives in the developers' speech, such as a [p] or [b]) which sometimes also needed some stretching by yet a different factor.

For a reproducible production from the raw material involving the various synchronization steps, we use the open-source video scripting system AviSynth.[18] An AviSynth script is executed by the AviSynth frameserver which then serves compatible programs such VirtualDub[19] with video frames and audio samples. These can either be played back in realtime (at least for simple scripts) or be sent to an encoder such as x264[20] to obtain a self-contained video file. Apart from being reproducible, video production through AviSynth did not put constraints on the video dimensions (that is, no constraints on top of the H.264 standard)—unlike Camtasia Studio which up to and including version 8 only supported a maximum resolution of 2048 by 2048 pixels.

Overall, the production of a single session video in the last setup would take at least one day of work with many iterations of manual finetuning.

17. Dictaphone model: Olympus VN-8700PC; Microphone model: AV-JEFE TCM 141

18. Homepage: http://avisynth.nl/index.php/Main_Page

19. Homepage: http://www.virtualdub.org/

20. Homepage: https://www.videolan.org/developers/x264.html

Appendix B.
Workshop Report for Company C

BERICHT

WORKSHOP PAAR PROGRAMMIERUNG

bei [redacted]

05.05-09.05. 2008

Laura Plonka
Freie Universität Berlin
AG Software Engineering

1 Einleitung und Motivation

In der Paar Programmierung (PP) arbeiten zwei Entwickler gemeinsam an einem Rechner an der Lösung eines komplexen Problems. Dabei sind sie gemeinsam für das entstehende Artefakt verantwortlich. Derjenige, der die Maus und Tastatur bedient wird als *Driver* bezeichnet, der andere als *Observer*. Die Rollen können jederzeit gewechselt werden. Obwohl immer nur einer *Driver* ist, sind beide Partner stets aktiv an dem Entwicklungsprozess beteiligt.

Während der Paar Programmierung (PP) arbeiten die Entwickler meist sehr konzentriert. Dabei bleibt wenig Möglichkeit den eigenen PP-Prozess bewusst wahrzunehmen und zu reflektieren.

Also wie kann man Paar Programmierer individuell in der Weiterentwicklung ihres PP-Prozesses unterstützen?

Genau an diesem Punkt setzt dieser Workshop an. Im Rahmen der Forschung der AG Software Engineering der Freien Universität Berlin (FUB) wurde ein Reflexions-/Feedback-Prozess für Paar Programmierer entwickelt. Ziel des Prozesses ist den Paar Programmieren die Möglichkeit zu bieten, den eigenen PP-Prozess besser zu reflektieren und zu analysieren. Dadurch können die Entwickler ihr Verhalten und ihren PP-Prozess individuell weiterentwickeln.

Hierfür zeichnen wir die PP-Sitzung eines Paars in ihrem Arbeitsumfeld bei [redacted] auf, analysieren die Daten und diskutieren die Ergebnisse am nächsten Tag anhand des aufgezeichneten Materials (ca. eine halbe Stunde lang) mit dem Paar.

Themen: Was lief gut? Was lief nicht gut? Warum?

Was könnte man am Verhalten ändern wollen?

2 Durchführung des PP-Workshop

Im folgenden Abschnitt wir die Vorbereitung sowie der Ablauf des Workshops beschrieben.

2.1 Vorbereitung

Vor dem Beginn des Workshops wurden die folgenden Punkte bearbeitet:

- **Einrichten der Infrastruktur**

  Um die PP-Sitzungen zu analysieren, müssen Daten über den Ablauf der Sitzung erhoben werden. Hierfür wurde das Aufzeichnungs-Setup auf einem Rechner bei [redacted] installiert, an dem die Paare im Laufe der Workshop-Woche aufgezeichnet wurden. Es erfolgte die Installation von Camtasia [1] (Aufzeichnung von Desktop, Ton und Webcam), sowie ein Webcam-Treiber. Damit die Entwickler weiterhin wie gewohnt auf ihren Entwicklungsrechnern arbeiten konnten, wurde eine Remote-Desktop Verbindung zwischen dem Aufzeichnungsrechner und den Entwicklungsrechnern hergestellt. Auf den Entwicklungsrechnern wurde zusätzlich ein Eclipse-Plugin (ECG [2] für das Logging von relevanten Computer-Interaktionen) installiert. Abbildung 1 zeigt schematisch den Aufbau der Infrastruktur. Der Aufzeichnungsrechner stand im Büro, in dem auch sonst gearbeitet und programmiert wird.

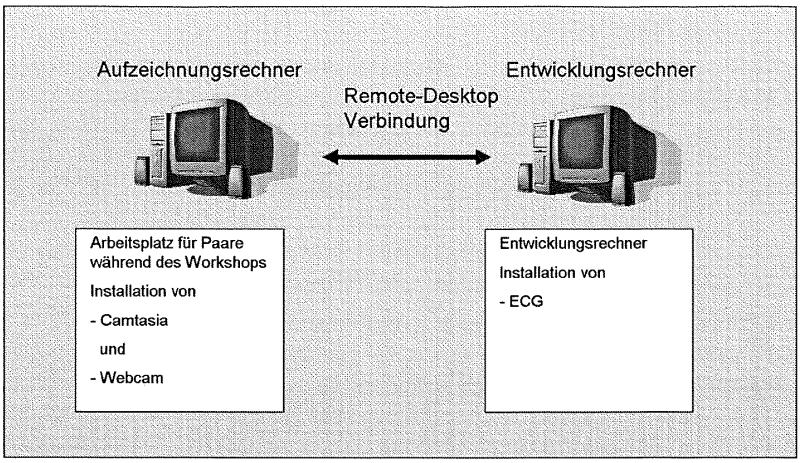


Abbildung 1: Schematische Darstellung der Infrastruktur, die Entwickler saßen in der Workshop-Woche, wenn sie sich aufzeichnen lassen wollten am Aufzeichnungsrechner, arbeiteten aber wie gewohnt auf ihrem Entwicklungsrechner

- **Entwurf Geheimhaltungsvereinbarung und Einverständniserklärung**

Um den Umgang mit den erhobenen Daten zu regeln, wurde von [redacted] eine Geheimhaltungserklärung entworfen. Die FUB entwarf ihrerseits eine Einverständniserklärung für die einzelnen Paar Programmierer, die sicherstellt, dass sie Programmierer wissen, welche Daten erhoben werden und freiwillig an der Aufzeichnung teilnehmen.

2.2 Einführungsveranstaltung

Zu Begin des Workshops wurde eine Einführungsveranstaltung von der FUB für die Entwickler von [redacted] durchgeführt. Ziel war, die Entwickler über den Nutzen und die Motivation, sowie den Ablauf zu informieren. Zusätzlich wurde vorgestellt, welche Daten von den Entwicklern erhoben werden. Diese Einführungsveranstaltung dauerte ca. 1 Stunde.

2.3 Datenaufzeichnung

Die Aufzeichnung von PP-Sitzungen war an dem vorbereiteten Aufzeichnungsrechner möglich. Es wurde jeweils ein Paar pro Vormittag bzw. Nachmittag aufgezeichnet. In einer Liste konnten die Paare eintragen, wann sie sich während der PP-Workshop Woche aufzeichnen lassen wollten.

Die Aufzeichnungsdauer pro Session wurde auf 1,5-2 Stunden beschränkt, es gab aber weder Einschränkungen bzgl. der Paarkonstellationen noch der Aufgaben, so dass die Paare während der Aufzeichnung ihren normalen Programmieraufgaben nachgehen konnten.

Während der Aufzeichnung wurden die folgenden Daten erhoben:

- Video der Programmierer mittels Webcam
- Audio der Programmierer über wireless Mikrophone
- Desktop der Programmierer
- Computer Log-Files (Loggen der Anwendung im Fokus, Dateiname, Anzahl der Editier-Schritte)

Die ersten drei Quellen wurden mittel Camtasia [1] aufgezeichnet, die Log-Files wurden mittels ECG [2] mitgeschrieben.

Zusätzlich wurden vor dem Start und nach dem Ende der Aufzeichnung von den Paaren jeweils einen kurzen Fragebogen (ca. 10 Min je Fragebogen) ausgefüllt.

2.4 Analyse der Sitzungen ohne die Programmierer

Im ersten Schritt der Analyse wurden die Daten für die Reflexions- und Feedback-Runde aufbereitet und von der FUB analysiert.

- Automatisierte Auswertung

  Die in der Sitzung erhobenen Daten wurden mittels eines von der FUB entwickelte Analysetools, *WhoTalks*, (eine genaue Beschreibung und Verwendung des Analysetools ist in [3] beschrieben) teil automatisiert ausgewertet und visualisiert. Das Computer-Log-File und die Ton-Spur werden automatisch ausgewertet und visualisiert. In der Analyse der Ton-Spur wird berechnet, wer wann und wie lange gesprochen hat, zusätzlich werden die Wechsel zwischen den Sprechern abgebildet sowie eine Statistik über die Verteilung der Sprachanteile. Dies ermöglicht eine einfache Analyse der Kommunikationsmuster (zum Beispiel: Monolog, Dialog) der Paare.

- Manuelle Videoanalyse

  Das Video (Desktop und Videobild der Programmierer) wurde zusätzlich noch manuell am Tag der Aufzeichnung von der FUB analysiert (Aufwand: ca. 2x Dauer des Videos . Dabei wurden zum Beispiel Informationen über das *Driver-Observer* -Verhalten sowohl quantitativ als auch qualitativ ausgewertet.

  In erster Linie dient die Videoanalyse der Identifikation von auffälligen Mustern und Verhaltensweisen des Paares. Diese Muster wurden im Rahmen der Analyse beschrieben und mit den Ergebnissen der Fragebögen in Beziehung gesetzt. Dabei wurden auch Informationen aus den Fragebögen in *WhoTalks* visualisiert.

Abbildung 2 zeigt den Prozess der Analyse mittels *WhoTalks*. Die resultierende Visualisierung (siehe Abbildung 3) und die Erkenntnisse aus der Analyse dienen als Einstieg für die Reflexions- und Feedbacksitzung.

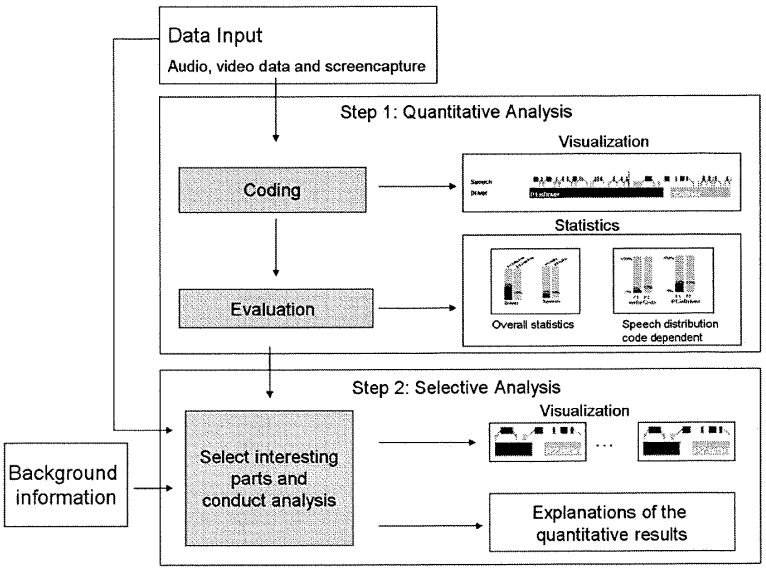


Abbildung 2: Schritte der Analyse; Im ersten Schritt werden die Daten quantitativ ausgewertet, die Evaluation ermöglicht einen statistischen Überblick und eine Visualisierung. Im zweiten Schritt werden die Videos qualitative analysiert und auffällige Muster bzw. Verhaltensweisen identifiziert und mit den Informationen aus den Fragebögen in Beziehung gesetzt und visualisiert.

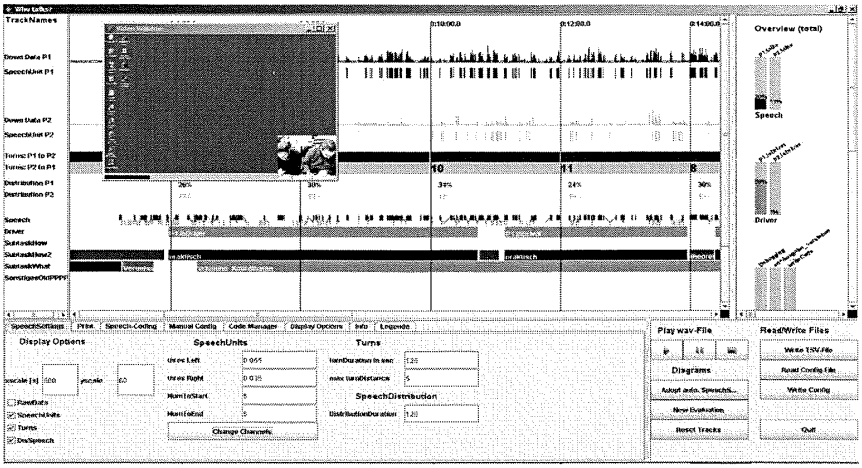


Abbildung 3: Screenshot einer Beispielvisualisierung von *WhoTalks*; im rechten Bereich sind die Statistiken abgebildet, der mittlere Bereich zeigt die Visualisierung der automatisiert berechneten Daten, die Visualisierung manuell kodierter Phänomene in ihrer zeitlichen Abfolge sowie das Video der Programmierer.

2.5 Reflexions- und Feedback-Runde mit den Programmieren

Die Reflexions- und Feedback-Runde fand jeweils am Tag nach der Aufzeichnung statt (mit Ausnahme des Freitags).

1. Fragen- und visualisierungsgestützte Reflexion

Anhand einer Liste von Fragen und durch die Visualisierung der Sitzung konnten die Paare den Verlauf der Sitzung im ersten Schritt nochmals nachvollziehen. Unter anderem wurden die folgenden Fragen von den Entwicklern diskutiert:

- Wie verlief die Sitzung?
- War etwas störend?
- Was war besonders hilfreich bei der Zusammenarbeit?
- Wie gut schätzen sie ihre Zusammenarbeit ein?
- Warum war es sinnvoll diese Aufgabe im Paar zu lösen?
- Wie schätzen Sie ihre *Driver*-Anteile ein?

In diesem Rahmen konnten Probleme, wie auch positive Verhaltensweisen von dem Paar erörtert werden. Häufig wurden im Zuge dieser Reflexion bereits Muster oder Verhaltensweisen von den Entwicklern angesprochen, die bereits während der vorab Video-Analyse aufgefallen waren.

2. Vorstellung der Ergebnisse der Videoanalyse: Externe Anregung und Feedback

Im zweiten Schritt erfolgte die Diskussion der Ergebnisse aus der Videoanalyse. Falls die Entwickler im ersten Schritt bereits die Aspekte thematisierten wurden diese mit dem externen Eindruck aus der Videoanalyse in Beziehung gesetzt und auch die externe Sicht diskutiert. Falls nicht alle Aspekte angesprochen wurden, die in der Video-Analyse identifiziert werden konnten, wurden die auffälligen Verhaltensweisen vorgestellt und mit dem Paar diskutiert.

Speziell wenn die Paare problematische Verhaltensweisen ansprachen, wurden Hinweise und Muster vorgestellt wie andere Paare solche Probleme lösen. In dem Abschnitt „Ergebnisse des Workshops“ sind die konkreten Aspekte beschrieben, die im Rahmen dieser Sitzungen diskutiert wurden. Insgesamt dauerte die Reflexions- und Feedback-Runde ca. eine halbe Stunde pro Paar.

2.6 Abschluss-Runde mit allen Programmieren

Zum Ende der Workshop-Woche wurde eine Abschluss-Runde mit allen Programmieren durchgeführt, um die Woche zu reflektieren. Die Themen waren:

- Vorstellung der Erkenntnisse aus dem Workshop
- Was hat der Workshop [redacted] bzw. den einzelnen Entwicklern gebracht?
- Was kann an dem Reflexionsprozess (organisatorisch wie auch inhaltlich) verbessert werden?
- Was passiert im Anschluss des Workshops mit den Daten?

Die Ergebnisse der Abschluss-Runde sind in Abschnitt „Erkenntnisse aus der Abschluss-Runde" diskutiert.

3 Ergebnisse des Workshops

Während der PP-Workshop-Woche wurden sechs Paare in sechs verschiedenen Paarkonstellationen aufgezeichnet. Von den sechs Paaren verwendeten vier Paare jeweils zwei Mäuse und Tastaturen an dem Aufzeichnungsrechner, zwei Paare programmierten gemeinsam mit einer Maus und einer Tastatur.

Alle Programmierer (7 Entwickler und Entwicklungsleitung) aus dem Team nahmen mindestens einmal an der Aufzeichnung teil.

Die Auswertungen der vorab Fragebögen zeigt, dass die Paare ihre Zusammenarbeit als unterschiedlich gut einschätzten; vier der sechs Paare schätzten ihre Zusammenarbeit als gut bis sehr gut ein und zwei Paare als befriedigend bis ausreichend.

3.1 Nutzen der Paar Programmierung bei [redacted]

Es gibt eine Reihe von Studien in der Wissenschaft, die den Nutzen von Paar Programmierung untersuchen. Diese Ergebnisse sind aber je nach Unternehmen und Motivation für den Einsatz von PP nicht zwingend übertragbar. Daher wurde im Rahmen der Reflexions- und Feedbackrunde mit den Entwicklern besprochen, warum sie PP bei [redacted] einsetzen, mit folgendem Ergebnis:

- ***Breitere Wissensbasis***

  Die Wissensbasis (zwei Programmierer wissen mehr als einer) wird vergrößert, dadurch können ***Ideen, Alternativen und unterschiedliche Sichtweisen auf das Problem*** diskutiert und ein geeigneter Ansatz zur Lösung eines Problems gewählt werden. Des Weiteren ermöglicht die breitere Wissensbasis die ***Vermeidung von Fehlern*** und speziell bei weniger vertrauten Codestellen eine ***verbesserte Orientierung im Code.***

- ***Wissensaustausch***

  Der Wissensaustausch beschreibt die Weitergabe von Wissen zwischen den Programmieren. Die Programmierer lernen im Rahmen der PP-Sitzung voneinander. Speziell in den Situationen, in denen einer mehr über den Code oder ein bestimmtes Programmierkonstrukt weiß, kann der Partner von dem Wissen des anderen profitieren.

- ***Mehr Disziplin***

  Speziell bei Aufgaben, die sehr viel Disziplin verlangen, wie beispielsweise das Schreiben von Tests unterstützen sich die Programmierer gegenseitig dabei, die Aufgabe zu erledigen. Die Partner ermutigen, in dem sie sich entweder aktiv auffordern die Aufgabe zu Ende zu führen oder aber einer der Partner einfach weiter daran arbeitet und somit der andere eine positive Art des Gruppenzwangs spürt.

- ***Bessere Konzentration***
  Die Zusammenarbeit fördert die ***Konzentration*** und dadurch auch einen ***besseren Flow***. Dadurch wird auch das ***gemeinsame Lernen*** (wenn sich beide bisher mit einem bestimmten Themengebiet nicht auskennen) vereinfacht.

- ***Mehr Spaß***
  Die Entwickler haben mehr Spaß an ihrer Arbeit und sind zufriedener.

3.2 Erkenntnisse aus den Feedback-Runden

Im folgenden Abschnitt werden die Themen und Erkenntnisse aus den Feedback-Runden anonymisiert vorgestellt, so dass kein Rückschluss auf einzelne Programmierer möglich ist.

Die Beschreibung und Diskussion der Themen erfolgt nicht quantitativ, d.h., die Ergebnisse werden nicht statistisch ausgewertet oder die Häufigkeit des Auftretens einzelner Aspekte gezählt, sondern die Aspekte werden qualitativ beschrieben.

3.2.1 Paarspezifische Aspekte

Im Folgenden werden Aspekte vorgestellt, die im Rahmen der Feedback-Sitzung von den Entwicklern diskutiert wurden und die den Paar Programmierungsprozess direkt betreffen.

- ***Driver-Observer Verhalten***

Allein die *Driver-Observer*-Verteilung (wer war wie lange *Driver*) lässt keinen Aufschluss auf Zusammenarbeit im Paar zu. So gab es Paare, die mit der Rollenverteilung zufrieden waren, die eine sehr ausgeglichene Rollenverteilung hatten als auch Paare, bei denen ein Partner den Großteil der Sitzung als *Driver* arbeitete.

Aber die Kombination aus der Rollenverteilung und der Art des Wechselverhaltens (Wechsel der Rollen *Driver* und *Observer*) war ein entscheidender Faktor. Bei Paaren, die mit der Rollenverteilung nicht zufrieden waren und nicht gut eingespielt waren (Information aus dem Fragebogen), verlief das Wechselverhalten häufig ohne Kommunikation. Dabei war einer der Partner meist sehr dominant und griff sich die Tastatur und Maus sehr schnell. Der weniger dominantere Partner versuchte zwar häufig die Mause zu greifen, war aber immer nur sehr kurz *Driver*, bevor der dominantere Partner wieder aktiv nach der Tastatur und Maus griff. Auffallend war auch, dass der dominantere Partner die Maus und Tastatur auch dann nicht losließ, wenn er sie gerade nicht verwendete (eine Art Reservierungsverhalten).

Sehr gut eingespielte Paare (Information aus dem Fragebogen) Paaren waren zufrieden mit ihrer Rollenverteilung. Bei ihnen verlief der Wechsel in einigen Fällen ebenfalls ohne Kommunikation, aber in gegenseitigem Einverständnis (häufig nachdem eine Teilaufgabe erledigt wurde), in anderen Fällen gab es einen Wechselprozess, in dem sich die Partner aktiv über den Wechsel absprachen, z.B. „Möchtest Du das machen?“, “ Soll ich mal wieder?”.

- ***Pausen***

Während der PP-Sitzungen arbeiten die Entwickler sehr konzentriert und fokussiert. Obwohl immer zwei Programmierer und somit zwei verschiedene Sichten auf das Problem vorhanden sind, kann diese Fokussiertheit auch dazu führen, dass das Paar gemeinsam an einem (eventuell suboptimalen) Ansatz festhält. Eine Pause kann dazu führen, dass das Paar die Aufgabe noch einmal auf einer konzeptionelleren Ebene betrachtet und sich später für einen alternativen Ansatz entscheidet.

- ***Arbeitsplatz und Rechnerwahl***

Im Rahmen des Workshops trat der Aspekt der Arbeitsplatzwahl nicht auf. Dennoch wurde von den Entwicklern auch angesprochen, dass die Arbeitsplatzwahl bzw. die Rechnerwahl Einfluss auf die PP haben kann. Denn jeder Entwickler konfiguriert seinen Rechner und vor allem die Entwicklungsumgebung individuell. Dies kann dazu führen, dass der Partner, an dessen Rechner nicht gearbeitete wird, sich eventuell nicht so gut zurechtfindet oder er beginnt die Konfigurationen anzupassen und somit die Einstellungen des anderen zu verändern.

- ***Blickwinkel auf den Monitor***

In der PP sollten beide Entwickler eine gute Sicht auf den Monitor haben. Dies ist allerdings oft schwierig durchzusetzen, da die Entwickler zum Teil vor dem Rechner hin und her rücken, um die Mause und Tastatur zu bedienen und dies häufig dazu führt, dass der *Observer* einen schlechten Blickwinkel auf den Monitor hat. In der PP gibt es den Ausspruch „Don't move the people, move the things!". Es kann ebenfalls zu einem schlechten Blickwinkel kommen, wenn die beiden Entwickler eine unterschiedliche Auffassung von einem einzuhaltenden Körperabstand haben. Dies kann dazu führen, dass eine Person, die einen größeren Abstand bevorzugt, langsam immer weiter weg rückt bis die andere Person mitten vor dem Monitor sitzt. Bei [redacted] wurde eine einfache Lösung für dieses Problem gefunden (funktioniert nur beim Einsatz von zwei Mäusen/Tastaturen). Zwischen die beiden Tastaturen wurde ein Stift in die Mitte des Monitors auf den Tisch geklebt, so dass die Tastaturen nicht verschoben werden können.

Als eines der wichtigsten Themen wurde das Kommunikationsverhalten zwischen den Entwicklern diskutiert. Eine ausgeglichene Kommunikation in Form eines Dialoges der Partner ist für den oben genannten Nutzen von PP Voraussetzung. Die statistische Auswertung der automatisierten Sprachanalyse (in [3]) zeigt, dass die Kommunikationsanteile (% der gesprochenen Zeit in Relation zur Gesamtlänge der Sitzung für die einzelnen Programmierer) bei allen Paaren ziemlich ausgeglichen waren. Folgende Punkte wurden im Bezug auf die Kommunikation genauer diskutiert:

- ***Entscheidungsfindungsprozess***

In der Paar Programmierung ist der *Driver* derjenige, der tippt. Dennoch sollten die Entscheidungen über das Vorgehen gemeinsam getroffen werden und für beide Partner stets nachzuvollziehen sein.
Dies ist ein Aspekt, der häufig in den Sitzungen angesprochen wurde.
Die Paare die ihre Zusammenarbeit auch im Nachhinein als sehr positiv beschreiben, trafen auch ihre Entscheidungen gemeinsam. Zusätzlich sicherten sich einige Paare durch regelmäßige Feedback-Fragen wie „Meintest Du das so?" oder "Ist das in Ordnung so?" ab, nachdem sie etwas geschrieben oder wiederholt hatten. Damit vergewisserten sie sich, dass der Partner ebenfalls die Entscheidung unterstützt.
In den Fällen, in denen es Probleme bei der gemeinsamen Entscheidungsfindung gab, traten solche Fragen weniger auf, bzw. in einem Stadium, in dem der Partner nicht mehr reagieren konnte, weil er bereits vom *Driver* abgehängt war. Der *Driver* verfolgte in dieser Situation seine eigene Idee ohne diese zur Diskussion mit dem *Observer* zu stellen oder den *Observer* darüber überhaupt darüber zu informieren. Der *Observer* wurde dann sehr passiv.
Dieses Problem war besonders schwerwiegend, wenn der *Driver* sich in dem Themengebiet besser auskannte. Speziell in dieser Situation sollte der *Driver* den *Observer* stark mit einbeziehen, um den **Wissensaustausch** zu fördern.
Eine andere Lösung ist, wenn der *Observer* aktiv nachfragt, sobald er merkt, dass er den Lösungsansatz nicht mehr nachvollziehen kann.
Auch dieses aktive Nachfragen des *Observer*s trat innerhalb von PP-Sitzungen bei [redacted] auf.

- ***Diskussionen auf der Meta/-Beziehungsebene***

Die Kommunikation der Partner ist der wichtigste Faktor in der Zusammenarbeit. Kommt es auf der Beziehungsebene zu Differenzen, so kann dies dazu führen, dass die Kommunikation gestört ist oder sogar, dass diese Differenzen auf die sachliche Ebene (z.B. Codeebene) übertragen und dort diskutiert werden. Im Rahmen der Feedback-Analyse gab es Paare, bei denen diese Probleme auftraten.
Sehr gut eingespielte Paare konnten solche Probleme auf der Metaebene direkt ansprechen und somit lösen. Bei weniger eingespielten Paaren führten solche Differenzen dazu, dass die Partner nicht mehr richtig auf die Vorschläge des anderen eingingen.
In der Reflexions- und Feedback-Runde kam heraus, dass **Zeitdruck** eine Diskussion auf der Metaebene hemmen kann. So gab es Entwickler, die berichteten, dass sie in Situationen ohne Zeitdruck eher das klärende Gespräch suchen.

- ***Phasen ohne bzw. mit weniger Kommunikation***

In der Visualisierung war zu erkennen, dass es zwischendurch in den Sitzungen Phasen gibt, in denen wenig bis gar nicht kommuniziert wurde. Diese Phasen traten typischerweise auf, wenn der *Driver* gerade Code schrieb. In der Diskussion mit den Paaren kam heraus, dass solche Phasen trotz fehlender Kommunikation nicht ineffektiv sind oder sich negativ auf die PP auswirken. Wenn eine Entscheidung über das weitere Vorgehen getroffen wurde, so kann der *Driver* die nächsten Schritte ohne zu kommunizieren umsetzen, da der *Observer* weiß, was gemacht wird. Dies führte zu der Frage, ob es in diesen Situationen nicht sinnvoll wäre, dass das Paar sich für gewisse Phasen trennt. In den Gesprächen mit den Paaren kam jedoch heraus, dass eine Trennung sich erst für wirklich lange Phasen (min. eine Stunde) lohnt, da sonst der Einarbeitungsaufwand für den Partner, damit er wieder mitdiskutieren kann, zu hoch sei. Abgesehen davon, sorgt der *Observer* auch in solchen Phasen für Fehlervermeidung.

3.2.2 Organisatorische Aspekte

Während der Reflexions-Feedback-Sitzungen wurden Themen diskutiert, die die Paarprogrammierung beeinflussen, allerdings nicht einer individuellen sondern einer organisatorischen Ebene zuzuordnen sind.

- ***Störung durch Dritte (direktes Ansprechen)***

Drei Paare wurden während der PP-Sitzung durch Dritte gestört, die mit Fragen oder Probleme auf einen der beiden Programmierer zukamen. Die Programmierer widmeten sich dann der Beantwortung der Fragen. Dies führte allerdings bei zwei Paaren dazu, dass sie in der Konzentration und in ihrem Flow unterbrochen wurden.

- ***Störung durch Dritte (Lärm)***

Einzelne Programmierer beginnen früh mit ihrem Arbeitstag. In einer Sitzung wurde im Hintergrund gesaugt, so dass das Paar nicht mehr in Zimmerlautstärke mit einander kommunizieren konnte. Dies führte ebenfalls dazu, dass sie in der Konzentration und in ihrem Flow unterbrochen wurden.

- ***Störung aufgrund technischer Probleme***

Technische Probleme treten im Bereich der Programmierung immer mal wieder auf, allerdings beeinflussen diese Probleme die Konzentration und den Flow der Paare. Technische Probleme traten in Form von Eclipse oder auch SVN-Problemen auf.

- ***Konflikt zwischen in individuellen und Teamaufgaben***

Einzelne Programmierer würden PP gerne häufiger einsetzen, allerdings haben alle Entwickler auch noch ihre individuellen Aufgabenbereiche. Dort entsteht der Konflikt

zwischen der Bearbeitung der individuellen und der Teamaufgaben. Da der Erfolg/Misserfolg der Bearbeitung der individuellen Aufgaben direkt auf den einzelnen Mitarbeiter zurückgeführt werden kann, werden die individuellen Aufgaben bevorzugt und somit weniger PP durchgeführt.

- ***Synchronisation der Entwickler***

Die Paar Programmierung setzt voraus, dass zwei Programmierer sich für eine PP-Sitzung verabreden. Die Synchronisation der Termine verlief in der PP-Workshop-Woche positiv. Dennoch wurde im Rahmen der Feedback-Runde auch angesprochen, dass unter anderem aufgrund flexibler Arbeitszeiten eine explizite Absprache notwendig und zeitweise auch schwierig ist.

- ***Coding/ Working Standards***

Nicht alle Programmierer können oder wollen miteinander programmieren. Dies liegt unter anderem daran, dass das Vorgehen für die Entwicklung nicht genau festgeschrieben ist. Die Programmierer entwickeln nach ihren persönlichen Präferenzen (beispielsweise test-first). Die Entwickler müssen sich daher untereinander über das Vorgehen einigen. Dies kann ggf. zu Problemen führen.

Bemerkung: Im Rahmen der PP-Workshop-Woche ist dieses Problem innerhalb einer Session nicht aufgetreten (ein solches Paar hat nicht gemeinsam programmiert), der Aspekt wurde aber dennoch im Rahmen einer Feedback-Sitzung angesprochen.

3.3 Erkenntnisse aus der Abschluss-Runde

3.3.1 Nutzen des Workshops

In der Abschlussrunde wurde der Nutzen des Workshops diskutiert. Für alle Entwickler bei [redacted] war die Reflexion eine sehr positive und interessante Erfahrung, die ihnen ermöglichte sich über ihre positiven wie auch zum Teil problematischen Verhaltensweisen in der PP bewusst zu werden und diese im Anschluss zu verändern und somit effektiver zusammenzuarbeiten. Speziell das externe Feedback von der FUB auf den PP-Prozess war für die Entwickler mit neuen Einsichten verbunden.

Die Entwickler wurden sich dabei der folgenden Aspekte bewusst:

- Welche Probleme treten in meinem PP-Prozess auf?
  - Wie kann ich diese beheben?
- Welche positiven Verhaltensweisen treten in meinem Prozess auf?
- Wie nimmt mein Partner das Programmieren mit mir wahr?
- Was schätze ich an meinem Partner?
- Warum setzte ich PP ein? Was bringt mir das?
- Welche äußeren Faktoren beeinflussen meinen PP-Prozess?

Da in der Reflexions- und Feedback-Runde individuelle Themen der einzelnen Paare besprochen wurden, konnten die Entwickler unterschiedliche Erkenntnisse aus der Runde mitnehmen. Allgemein können aber die folgenden Richtlinien (ohne organisatorische Erkenntnisse) für den PP-Prozess abgeleitet werden:

- Kommuniziere mit dem Partner
- Diskutiere Entscheidungen
- Teile Dein Wissen
- Greif Dir nicht die ganze Zeit die Maus und Tastatur – der Partner möchte auch in Ruhe tippen

- Etabliere einen aktiven Wechselprozess (für weniger eingespielte Paare)
- Mach Pausen
- Sprich Probleme offen an

Abgesehen von dem Nutzen für die einzelnen Entwickler hat der Workshop die Motivation PP einzusetzen erhöht. Zusätzlich wurde der Mehrwert, den PP für [redacted] bietet, von externer Seite beleuchtet. Außerdem ermöglicht der Workshop Einblicke in die Herausforderungen, mit denen sich das Team auseinandersetzen sollte, um PP möglichst effektiv umzusetzen (siehe Abschnitt Organisatorisch Aspekte).

Bislang konnte der langfristige Nutzen des Workshops noch nicht evaluiert werden. Hierfür wird noch ein Fragebogen an die Entwickler ausgegeben. Zusätzlich wäre es sinnvoll einen solchen Workshop zu wiederholen, um die langfristigen Veränderungen zu untersuchen.

3.3.2 Feedback zur Organisation des Workshops

Das Feedback zur Organisation war sehr positiv. Speziell, dass die Aufzeichnung für die einzelnen Programmierer wenig zusätzlichen Aufwand bedeuteten und dass die technische Infrastruktur gut funktionierte. Es gab allerdings auch Anregung zur Verbesserung:

- Längere Aufzeichungsdauer

Die Länge der PP-Sitzungen war auf 1,5 bis 2 Stunden begrenzt. Einige Programmierer setzte diese Einschränkung unter Zeitdruck. Daher sollte die Aufzeichnungsdauer flexibler gestaltet werden können.

- Selbstständiges Starten der Aufzeichnung

Vor jeder Aufzeichnung mussten die Programmierer den Start mit der FUB absprechen. Von Seiten der Programmierer wäre es allerdings wünschenswert, die Aufzeichnung ohne Absprache starten zu können.

- Einsatz des Stativs für die Webcam

Die Webcam war auf einem Stativ befestigt, dass auf dem Schreibtisch des Aufzeichnungsrechners stand. Es wäre sinnvoll ein größeres Stativ auf den Boden zu stellen.

- Fokus der Webcam

Positiv wurde angemerkt, dass das Sichtfeld der Webcam hauptsächlich die Tastatur und die Maus und dadurch die Hände der Entwickler aufzeichnete nicht schwerpunktmäßig die Gesichter. Dadurch fühlten sich einige Programmierer weniger beobachtet. Das bewusste Wahrnehmen der Webcam variierte zwischen den Entwicklern jedoch sehr stark. Einige fühlten sich sehr beobachtet (das kann auch das Verhalten beeinflussen) andere nahmen die Webcam nach den ersten 5 Minuten nicht mehr wahr.

- Größerer Schreibtisch

Für die PP sollte ein großer Schreibtisch zur Verfügung gestellt werden, so dass beide Entwickler ausreichend Platz haben.

- Sehr begrenzte Anzahl von Aufzeichnungsmöglichkeiten

Insgesamt wurden 6 Paare aufgezeichnet. Allerdings hätten die Programmierer gerne häufiger an der Reflexions- und Feedback-Runde teilgenommen. Da eine Sitzung nicht zwingend repräsentativ ist und die Programmierer ihr Verhalten gerne in unterschiedlichen Paarkonstellationen reflektiert hätten. Als angemessener Zeitraum wurden zwei Wochen vorgeschlagen bzw. eine Wiederholung des Workshops.

4 Zusammenfassung und Ausblick

Im Zuge des Workshops wurden sechs Paare aufgezeichnet und analysiert. Die Entwickler reflektierten über ihre Verhaltensmuster beim PP. Für die Entwickler war der Workshop eine sehr interessante und positive Erfahrung.

Bei der Reflexion und Analyse wurden positive, wie auch negative Verhaltensmuster im PP sowie externe Einflussfaktoren identifiziert und diskutiert.
Speziell bei der Diskussion der paarspezifischen Aspekte konnten die Entwickler Aufschluss über ihr eigenes Verhalten bekommen sowie Anregungen zu mögliche Verbesserungen ihres PP-Prozesses durch Hinweise auf alternative Verhaltensmuster bekommen.
Die Identifikation der organisatorischen Herausforderungen wird das Team unterstützen, das Umfeld für die Paar Programmierer weiter zu verbessern.

Derzeit erfolgt eine Detailanalyse der Videos, in der die oben genannten Aspekte genauer im Videomaterial nachvollzogen werden. Sobald aus dieser Analyse Ergebnisse vorliegen, werden diese an [redacted] weitergegeben.

Zusätzlich erfolgte bereits ein Vergleich von Studierenden mit den [redacted] Paar Programmierern [3]. Dabei wurden im Schwerpunkt die *Driver-Observer*-Verteilungen, sowie die Sprachverteilung untersucht. Das Ergebnis dieser Studie zeigt, dass sich bereits anhand des Vergleiches der quantitativen Merkmale Unterschiede zwischen Studenten und den [redacted] Programmieren zeigen. Die vollständige Studie ist dem Bericht beigelegt.

Sehr interessant wäre auch die Durchführung eines weiteren Workshops, um unterschiedliche Paarkonstellationen zu untersuchen und die Auswirkungen des ersten Workshop zu evaluieren.

.

5 Referenzen

[1] TechSmith Corporation. Camtasia studio 4.0.1. http://www.techsmith.com.
[2] F. Schlesinger and S. Jekutsch. ElectroCodeoGram: An environment for studying programming. Workshop on Ethnographies of Code, Infolab21, Lancaster University, UK, March, 2006.
[3] Laura Plonka. A comparison between Student and Professional Pair Programmers. Workshop on Psychology of Programming. In press.