A comparison between Student and Professional Pair Programmers

Laura Plonka[1]

Institut für Informatik, Freie Universität Berlin
plonka@inf.fu-berlin.de

Abstract. Most pair programming studies are conducted with student pair programmers. This might be due to the fact that it is easier for the researcher to record students. But often it is desirable to get information about professional pair programmers. Could we extrapolate the results from student studies to the professionals? In this paper I present a comparison of student and professional pair programming sessions in order to reveal differences and similarities. The analysis focuses on the speech contributions and the driver distributions of the pairs. We show that professional pairs are balanced regarding speech distribution within the pairs, whereas some student pairs are very unbalanced. For the driver distributions, no typical pattern could be found for both groups.

1 Introduction

Pair Programming (PP) is a software engineering practice in which two people share one computer, taking turns in their use of keyboard and mouse and cooperate closely throughout the session. Supporters of PP claim a number of benefits from using this practice compared to solo programming, for example, knowledge transfer among the development team, increasing quality of code, higher concentration of the programmers or decreased error rate.

In order to analyze the benefits and the process itself, PP has been the subject of many empirical investigations in the last few years [2, 3, 8–10, 14–17]. Often it is desirable to learn about the PP process in the professional development world in order to understand how PP works in the real world and furthermore for example, in order to provide information about how the professional pair programmers can improve their own PP process.

But most of the studies or experiments have been conducted with student pair programmers as participants due to easier recording possibilities [16, 15]. Unfortunately student PP sessions usually differ at least in some aspects from professional PP sessions: The scope of the task in student settings is predefined by the researcher and normally not a real world or very complex task, because the students have a limited timeframe to solve the problem. The advantage of a predefined task is that the researcher can comprehend and evaluate the solution process of the students. In contrast, in the professional settings most researchers observe programmers in their daily development tasks with their daily work setting, whereas the students often work in an artificial setting. The latter conditions allow a better comparability and repeatability but do not represent the real world setting. Finally, the experience of programming as well as the experience of PP of professional programmers is usually higher.

Studies with student sessions are often considered to be of limited value, because the extrapolation of the results of an academic setting to a professional setting is not always possible [5]. It is apparent that the external circumstances, for example, task, experience of student and professional PP sessions are different, but does the PP process itself differ between student and professional sessions? Due to the fact that there are many studies with student pair programmers and that recording students is easier than recording professionals, it would be useful to know if we can extrapolate the results of the analysis of student PP sessions to professionals. In order to find out about the limitations of such an extrapolation, our research questions are: Are there differences in the behavior between students and professionals in the PP context? Could we perhaps find a filter in order to distinguish between student pairs who act more similar to professionals than others?

In order to answer these questions, I have compared undergraduate students with professional pair programmers in a quantitative analysis. As possible indicators I have chosen speech contributions and driver distributions within the sessions. Furthermore, I analyze exemplary parts of the session in greater detail in order to interpret the quantitative data.

I will first outline related works (section 2) on the driver-observer behavior and the communication patterns in student as well as in professional settings. Thereafter the paper describes the nature and origin of our raw data (section 3) and explains the steps of the analysis (section 4). Then I present the results (section 5) and finally close with our conclusions (section 6) and outlook (section 7).

2 Related Work

In the following section I summarize studies of PP in commercial and academic environments.

2.1 Studies in an Academic Setting

There are many studies investigating student pair programmers, most of the time in a quantitative manner in order to analyse the benefits [16, 6, 9, 11, 10]. We will not discuss these studies here, because I focus only on the studies conducted with students which investigate the driver behavior or the communication process.

Cao and Xu [3] investigate the activity patterns of PP for different pair constellations in a qualitative manner. They divided the programmers into low, medium, and high competence level programmers. In their analysis they have investigated the differences between various pair constellations. The high-high pairs enjoyed the PP experience more than the other pairs and had the highest interaction frequency, the medium-medium pair exchanged a very limited amount of knowledge, and in the high-low pair the interaction frequency was the lowest and the more expertised programmer took a distinct leader role.

In [7], Hfer focuses on the distribution of mouse and keyboard control of 9 student pairs. Therefore the control of the mouse or keyboard was defined by the time when a programmer had a hand on one of those devices including the time without any input activity. Hfer detects that the pairs have changed the control frequently, but the pairs did not share the control equally.

2.2 Studies in a Professional Setting

Bryant compares the interaction behavior in a professional setting depending on driver and observer roles, subtask and PP experience within professional pair programmers. In [2], Bryant studies the difference of interaction type and frequency in novice versus expert pair programmers in a professional environment. She found out that more expert pair programmers have a lower interaction frequency compared to novice pair programmers, especially the number of suggestions per session is less. Furthermore, she analyzed the interaction behavior while each programmer has the control of keyboard and mouse, with the result that the more experienced pair programmers seem to have a defined set of behavior independent of who is driving, whereas the behavior of the less experienced pair programmers seems to be driver-dependent. In another evaluation of professional pair programmers, she coded the contribution to the conversation (if the contribution added new information to the task) of each programmer, and generic subtasks [1]. Both programmers contribute almost the same amount of contributions to all subtasks, and driver and observer verbalize almost the same amount (the driver verbalizes slightly more). Moreover, in [12], Bryant studies the abstraction level of driver and observer. She states that the pair programmers did not discuss in general on different levels of abstraction according to their roles.

Those studies compare the interaction behavior in a professional environment dependent on driver and observer roles, subtask and PP experience, but there is no comparison between

students and professional pair programmers. Chong [4] conducted ethnographic observations with two development teams. She investigated the programmer interaction in dependency on the roles of driver/observer and the expertise of the programmers. She states that apart from typing there is no division of labor between the driver and observer. In pairs with the same level of expertise both seem to contribute at almost the same rates and with the same level of abstraction. But the keyboard control seems to influence the decision making to an advantage for the driver. In pairs with different expertise, the programmer with more expertise dominates the interaction.

3 Data Capturing

In this section I describe our recording setup and the data capture in the student as well as the professional setting. In order to compare student and professional pair programmers the recordings took place on the one hand in the typical setting for students and on the other hand in a company for the recordings of the professional developers.

3.1 Recording Setup

During the PP session, I have recorded the following data equally for student and professional pair programmers:

1. A stereo audio recording captured verbal communication between the participants,
2. a video of the programmers captured who is driver and observer,
3. a full-resolution screen recording captured almost all computer activities of the programmers on a fairly fine-grained level.

All data have been recorded with Camtasia Studio [13] into a single, fully synchronized video file in which the video image of the camera (recorded with a Logitech 5000 webcam) is superimposed semi-transparently onto a corner of the video of the screen so that all information is visible at once. For the stereo audio recording an external soundcard (Tascam USL122) in combination with two professional wireless microphones (audioTechnica 700 Series Professional UHF Wireless Systems) has been used.

In order to get as much background information as possible (e.g. PP experience, programming experience) about the participants, all pair programmers filled in questionnaires before and after the session.

3.2 Recordings of Student Sessions

The student recordings took place as part of an a exercise of a software engineering course for undergraduate students. The participation was voluntary and the students had chosen their partners for the PP session on their own. Ten programmers (nine men and one woman) participated in the recording. Each of the five pairs was recorded once. The recording of the session started with the handover of the exercise. All pairs worked on the same task. The task was a small algorithmic problem. The pairs had no time limit for the task. All pairs solved the task within an average work time of 1 hour. Eight of the student programmers had previous PP experience (maximum of two years). The average programming experience of the students is three years (refer to table 3.3).

In addition to the recordings the pairs filled in in questionnaires before and after the session.

3.3 Recordings of Professional Sessions

Furthermore, I recorded six professional pair programming sessions within one week in a company in the geographic information system industry. The development team (consisting of seven developer and the head of development) uses an adapted agile process, in which PP is applied regulary. In general the developers choose the partner for the PP sessions on their own. Some pairs prefer to use one mouse and one keyboard, other pairs prefer to use two mouses and two keyboards.

For the recorded PP session I limited the duration of the session to two hours. But in order to record the pairs in their accustomed professional settings and get data from the real world, I did not make any constraints about the task, the pair constellation or the setting of PP (one mouse and keyboard or two mouses and keyboards). Therefore the pairs worked on their daily tasks in the preferred PP setting and with a partner chosen on their own. All developers participated in the recordings.

For this paper I have analysed four pairs. In two sessions not all sources described above could be recorded due to recording problems of the microphones. Those pairs are not analysed yet. In the following description of the sessions and the experiences of the programmers, I will focus only on the four analysed pairs.

Six programmers (two women and four men) participated in the four sessions. Two pairs consisted of one woman one man, the other two pairs consited of two men. All four pairs had different pair constellations. Two of the four pairs worked with two mouse and two keyboards while pair programming. The average professional programming experience of the programmer is nine years (refer to table 3.3) and the average pair programming experience (PP experience as professional plus PP experience in education) is ten years and six month.

In addition to the recordings, the pairs filled in in questionnaires before and after the session. Moreover, an interview with each pair was conducted one day after the recording of the pair.

Table 1. Overview of the students and professional programmers background.

Background information	**Students**	**Professionals**
Average programming experience (in years)	3	9 (as professionals)
Total duration of analyzed sessions	6h 28min	6h 11min
Task	small algorithm	daily work
# Sessions	6	4
# Pairs using two mouses/keyboards	0	2

4 Analysis

For a quantitative comparison of professionals with student pair programmers it is necessary to evaluate as many pairs as possible. This is why I have chosen the speech contribution (the time of talking of each programmer) and the driver distribution as possible indicators for differences between student and professional pair programmers and have sped up the analysis through an automated analysis of the speech contributions (described in section 4.2). Unfortunately the coding of the driver needs to be performed manually on the video data, which is very time-consuming and limits the number of sessions.

The analysis process has two steps: First, a quantitative analysis was conducted, and second, in order to interpret the quantitative data, I use exemplary sessions or part of the sessions for a selective analysis. The whole analysis process is shown in figure 1.

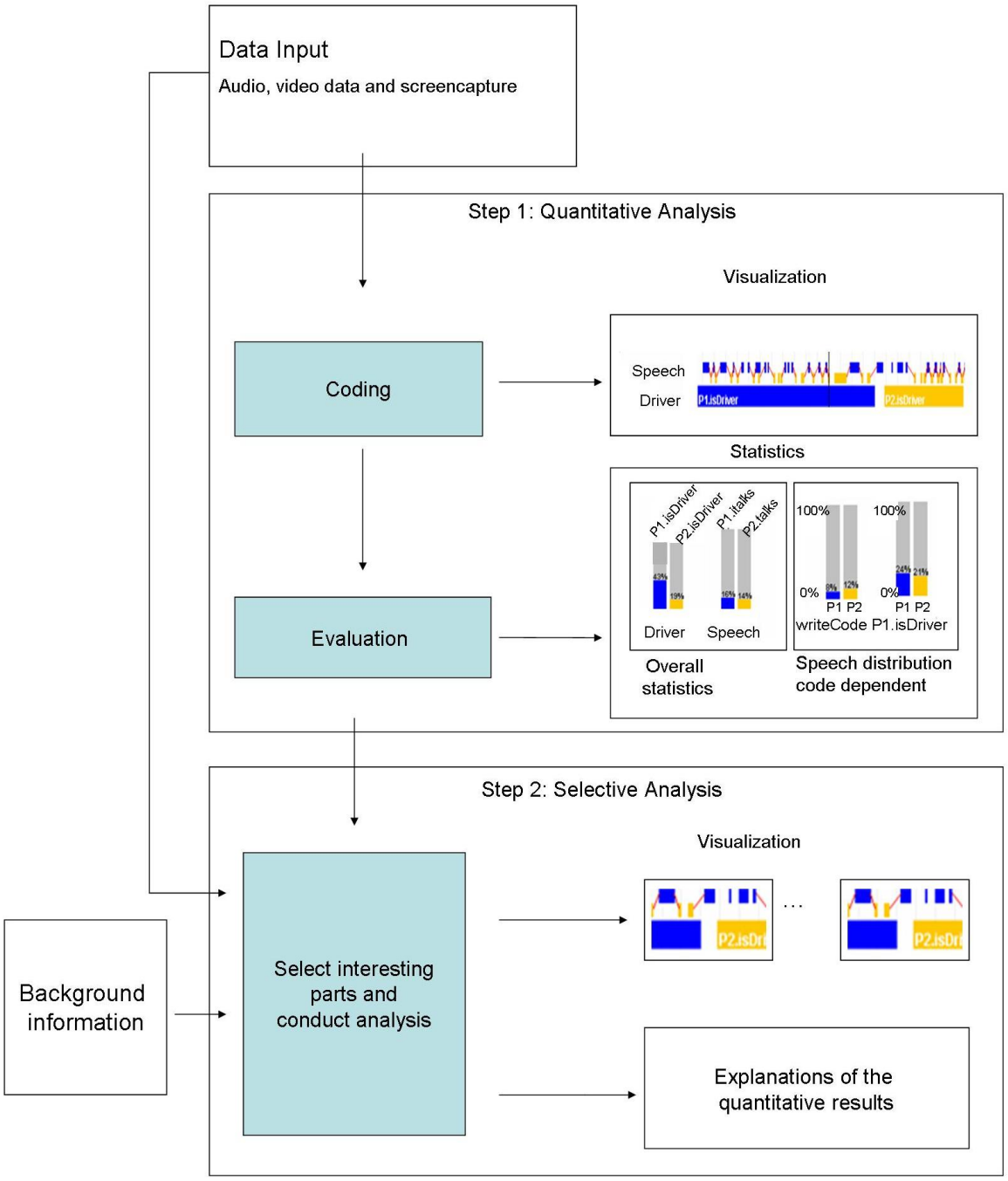

Fig. 1. The steps of the analysis; The first step is the quantitative analysis of the input data, the evaluation provides a statistical overview and a visualization. The second step starts with an analysis of exemplary selected data as video or background information (questionnaire/interview), which are analyzed, visualized if possible, and evaluated.

4.1 Coding

In order to get an overview and conduct a statistical evaluation of the driver distribution and communication behavior (in general and according to the roles of driver and observer) in the first step, I have coded all sessions with the following codes grouped into tracks (for visualization):

- *Speech-Track*: with the codes *P1/P2.talks* (programmer 1 talks, programmer 2 talks, respectively),
- *Driver-Track*: with the codes *P1.isDriver/P2.isDriver*.

The speech codes are necessary in order to determine the speech contributions of each programmer (the time each programmer is talking). All verbal contributions of the programmers are coded without considering semantic information. For the coding of the speech contributions I have used the automated speech analysis tool *WhoTalks* (refer to section 4.2). The driver codes are used for the analysis of driver and observer distributions. *isDriver* is coded when the mouse or the keyboard is used (key pressed or mouse moved) by one of the programmers. We have identified the movement of the mouse or keyboard input through the screen capture, and the current driver through the webcam video. In the sessions in which two mouses and keyboards were used, the observer and driver often changed roles for just one key press or a short mouse movement. Those situations are not considered as driver-observer changes due to the time-consuming analysis.

The results of the quantitative analysis are two different types of statistics: The first type offers an overview of the overall duration of the codes as a percentage of the whole duration of the PP session, sorted by tracks (e.g. the speech distribution in the results of the quantitative analysis (overall statistics) in figure 1). The second type of statistic combines the driver codes with the speech codes: It calculates the percentage of the speech units overlapping with driver codes (refer to figure 1: results of the quantitative analysis, the speech distributions while speaker 1 is driver). Furthermore, I visualize the data in order to get an overview of the chronological development of the speech contributions as shown in figure 1.

In a second step I selected different exemplary parts of the video data, the visualization or other background information depending on the results of the quantitative analysis and took a closer look at those data in order to understand the quantitative results. The findings of this step are helpful to interpret the quantitative data and provide further insights into the process of PP.

4.2 Tool Support during the Analysis

We used *WhoTalks* for speeding up the analysis process. *WhoTalks* is an analysis tool for video and audio data which was developed in our group for combined qualitative and quantitative studies with the main focus on the communication behavior in the PP research context. *WhoTalks* allows automated quantitative evaluation of the communication process, a manual coding process, statistical evaluation, and visualization of the data (refer to figure 2).

It supports our first step in the coding process by the automated analysis of the current speaker and the duration of the contribution (speech unit) from the stereo audio-file. Therefore WhoTalks seperates the stereo-audio track in two mono audio-channels and thereby it can distinguish the speaker, because each speaker belongs to one mono audio-channel. In each mono audio-channel the start of a speech unit is recognized when a user-defined number of consecutive samples of a specific channel are above a user-defined threshold, and the end when the same number of consecutive samples are below the same threshold. The threshold should be adapted to the volumes of the two speakers' voices. All calculated speech units are shown on the timeline below the visualization of the audio data (figure 2). The driver track was coded with *WhoTalks*, too. All annotations are visualized on the timeline. This visualization gives an overview and allows an easy recognition of interesting patterns and starting points for our second

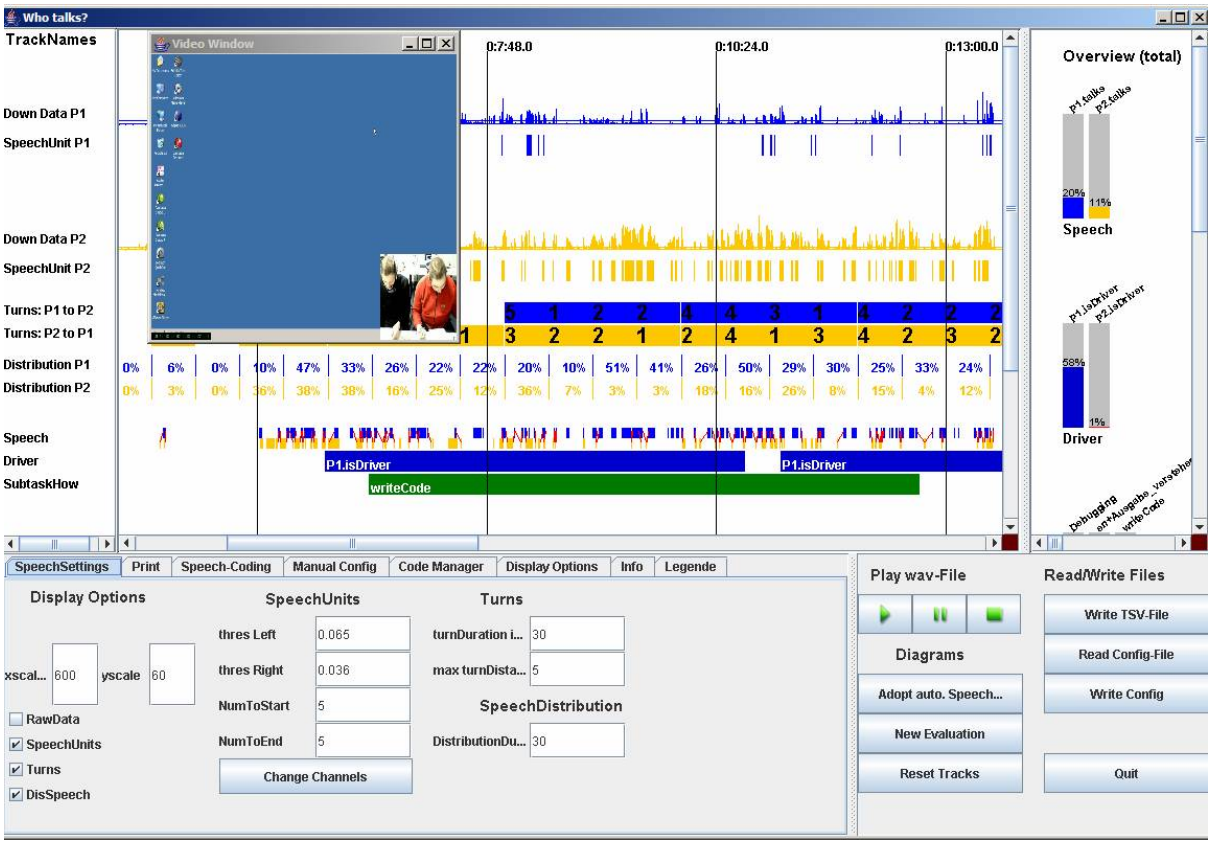

Fig. 2. Screenshot of WhoTalks: The left pane lists the tracknames, the center pane shows the visualization window with the timeline, speech units, and annotations. The video is played in a separate, resizable window. The right pane shows the statistical evaluation, and the bottom pane allows setting of the speech parameters.

analysis step. Furthermore, *WhoTalks* provides two types of statistical evaluation (described in section 4.1) of the coded data, which I have used for the quantitative comparison between the students and the professionals.

5 Results and Discussion

This section describes the results of the quantitative analysis and the findings from the selective coding step. Due to the fact that only ten pairs were analyzed, the quantitative analysis does not look for statistical significance. The student sessions are represented by S1 to S6 and the professional sessions by Pro1 to Pro4. The pairs S6, Pro3 and Pro4 are mixed pairs (one woman one men) the others are male teams.

5.1 Speech Contributions

First I have analyzed the speech contributions of the programmers. Figure 3 gives an overview of the speech contributions of each programmer in percent of the whole session duration: All programmers, except S5 and S6, contribute almost the same amount of speech during the sessions. Moreover, except for S5 and S6, the distribution of speech contributions within the sessions are almost balanced between the programmers. Sessions S5 and S6 have one programmer who contributes far more than the average amount of speech, and the distribution within these pairs is very unbalanced. In S5 the developer P1 has the higher amount of speech. Investigating the video of S5, the developer P1 verbalizes a large part of computer activities and propose the next actions.

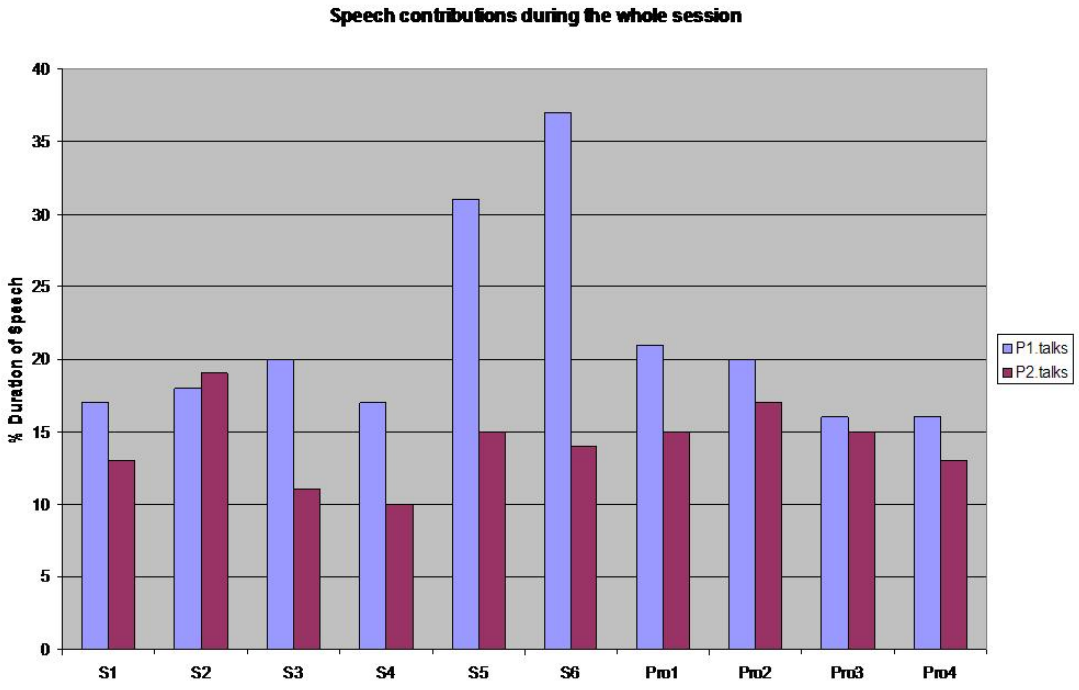

Fig. 3. Speech contributions as a percentage of the whole session duration.

Figure 4 shows an extract of 7 minutes and 30 seconds of the speech units of a balanced pair, Pro4, and an unbalanced pair, S6. It gives a good impression of the difference in the communication behavior: In Pro4 a lot of phases can be seen in which either nobody talks or the programmers have a dialog. In pair S6 the programmers conduct a dialog, too, but there are situations in which P1 (the programmer with more speech contributions) holds monologs which are sometimes answered with short comments of his/her partner and there is no phase longer then 10 seconds (except during the reading of the task) in which nobody talks. The evaluation of the visualization of S6 and Pro4 results in a different structure of the communication process.

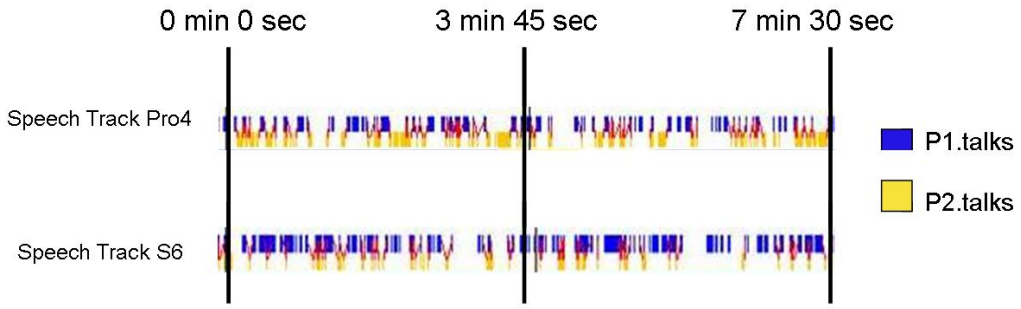

Fig. 4. An extract of 7 min 30 sec of the visualization of the pair Pro4 and pair S6: Pro4 conducts a dialog (the speaker often changes), P1 of pair S6 holds a monolog with short comments of his/her partner.

We propose the hypothesis that a monolog structure and an unbalanced distribution of speech contributions during a pair programming session do not occur in normal professional

pair programming sessions but probably in novice-expert situations (e.g. teaching a newbie). We evaluated the background information from the questionnaire of the programmers in pair S6 to investigate if the pair is a newbie-expert constellation, but they estimated themselves on almost the same programming level with different experience in programming (programmer 1: 2 years, programmer 2: 5,5 years). Furthermore, they had programmed in this constellation for half a year and described themselves as very well attuned.

5.2 Driver Distributions

Figure 5 shows the driver distribution during all sessions. Neither within the professional data nor the student data a pattern or regularity can be identified. In both types pairs appear with a balanced as well as unbalanced distribution of the driver role. But it is remarkable that the student pairs S3 and S4 do not change the roles, and the pairs S5 and S6, who are very unbalanced according to the speech contributions, are also unbalanced considering the driver distribution.

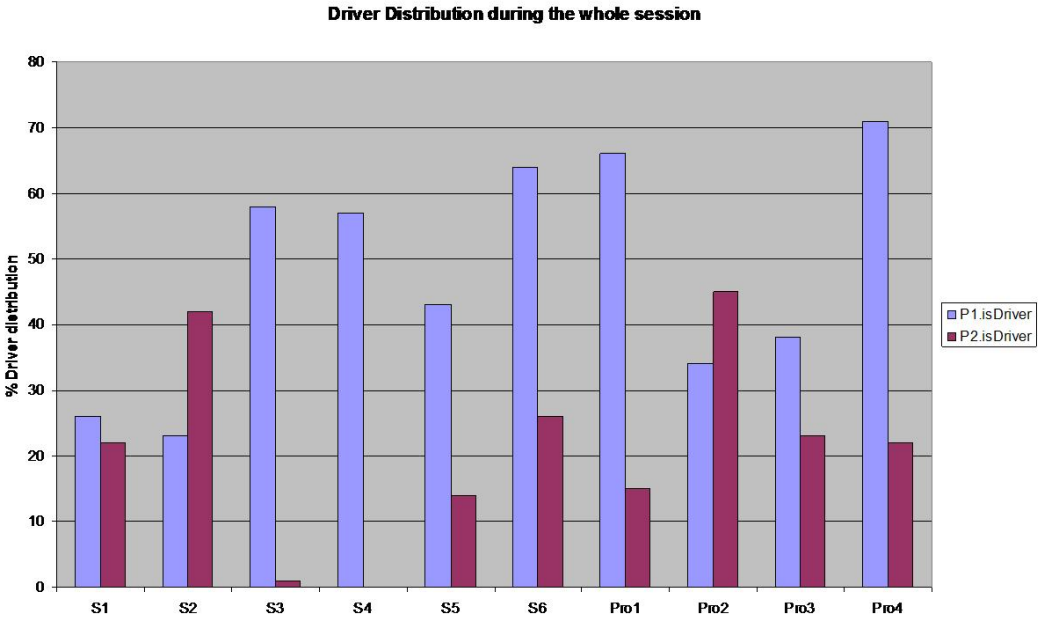

Fig. 5. Driver distributions during the whole session.

We used the video material in order to understand the behavior of pairs S3 and S4 and found out that these pairs had decided at the beginning of the session who will be the driver, with the motivation that in one case one of the programmers typed more quickly than the other and in the other case one of the programmers admitted that he had not been programming for a while. For the professional programmers I have analyzed data from the interview which I had conducted one day after the recording: Both pairs, Pro4 (unbalanced) and Pro2 (balanced), were content with their role distributions. In the constellation Pro1 (unbalanced), P2 was not satisfied with his driver part and would like to involve himself/herself more in the driver process.

The evaluation of only the driver distribution reveals no differences between professionals and students and no patterns within the two groups. We conclude that the driver distribution depends more on the personalities and characters of the programmers.

5.3 Combining Speech Contribution and Driver Distribution

The result of the quantitative analysis of the speech contributions overlapping with the driver codes is presented in figure 6. It shows for each pair the speech contribution of each programmer, while P1 and P2 is driving. Due to the fact that the pairs S3 and S4 do not change driver and observer roles, the pairs are not included in the further investigation.

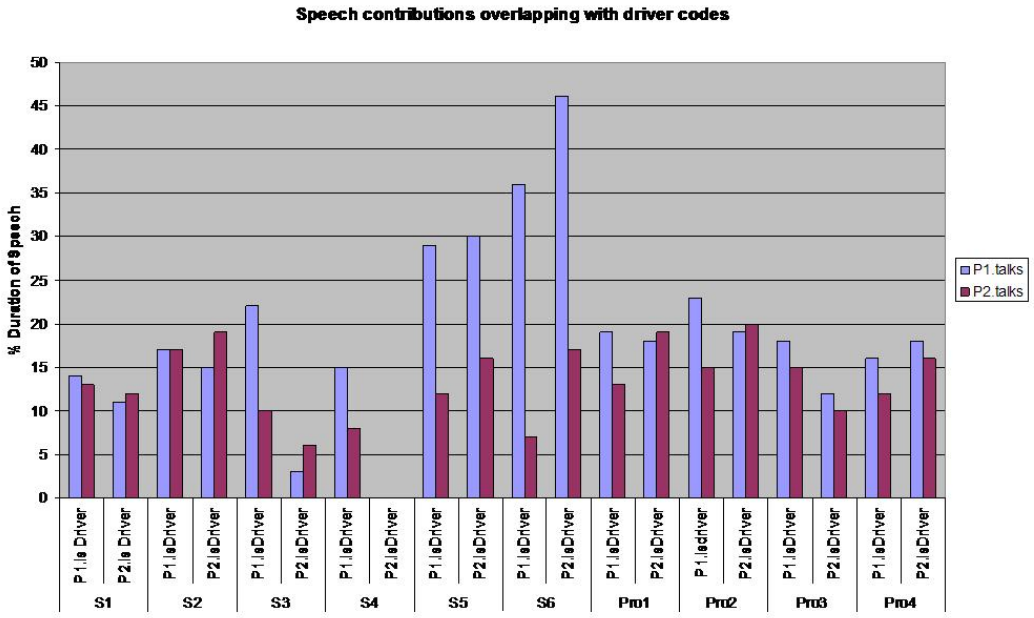

Fig. 6. Combining speech contribution and driver distribution.

The speech contributions of all professional pairs and two student pairs remain balanced independent of who is driver or observer. In the student pairs S5 and S6 (very unbalanced considering the driver distribution and the speech distributions), the programmer P1, who has a higher speech contribution and driver contribution talks twice as much as the partner, irrelevant of who is currently driving. For all balanced pairs it can be noticed that the programmer who is driving also talks slightly more than the observer, thus participating more actively in the communication. This is in accordance with findings presented in [1].

Further investigations of the video data of pair S5 revealed the communication pattern of those situations: While P1 is driving, he verbalizes all of his activities. While observing, he proposes the next actions, dictates word by word what to write or verbalizes the activity of the driver.

5.4 Limitations

The obvious limitation of this study is the small number of sessions and the origin of the data (four professional sessions of a single company and six sudent pair performing a single algorithmic task) which have been analyzed. In addition, the tasks of the professional programmers

were different, because they did their daily work task. The task might influence the process and the communication behavior. Moreover, the professionals used their preferred PP settings (two mouses, two keyboards or one mouse, one keyboard), because I tried not to influence their work settings with our study. In comparison to that all student pairs worked with one mouse and one keyboard. This could also be a factor which influences the PP process and the results of the study. Furthermore, only two quantitative aspects (driver distribution, speech contribution) of potential differences between students and professionals are used for the comparison of the whole sessions, and only single sessions or parts of sessions have been used for a deeper analysis. There are more, in particular qualitative, aspects which need to be evaluated, e.g. a semantic analysis of the communication, in order to understand the differences in detail.

6 Conclusion

We have compared student and professional pair programming sessions considering the aspects of speech contributions and driver distributions in a quantitative manner. In future work this may help to extrapolate findings from the analysis of student PP sessions to professional PP sessions. This is desirable because student sessions can be recorded more easily. Regarding the speech distribution of professional and student pair programmers, both groups seem not to differ in general. But there are student pairs where one of the programmers talks more than the average, thus displaying a very unbalanced distribution of speech among the programmers even if they estimated themselves to be on the same programming level. The unbalanced distributions are a result of a monolog structure which sometimes occurs in student pairs. We expect such a structure to occur in professional pairs only in an expert-newbie constellation. Therefore I suggest that the evaluation of speech distributions can act as a filter to sort out student pairs who do not act like professionals according to their interaction behavior.

The results of the analysis of the driver distribution show that there are no patterns within the two groups. Therefore there seem to be no differences between the driver distribution of students or professional pair programmers. In addition, I found out that student programmers who dominate the communication significantly have also significantly higher driving parts then their partner. This could be an indicator for a leader role in the pair, which has often been detected during PP analysis (e.g. [3]). Furthermore, I found no indicator that a satisfied pair needs to have a balanced driver distribution.

The combination of driver distribution and speech contribution indicates that the speech contribution of professional pair programmers remains balanced irrelevant of who is currently driving. The same pattern was detected in two student sessions. In two other student pairs (S5, S6: very unbalanced considering the driver distribution and the speech distributions) the 'leader' student seems to dominate the communication irrelevant of who is driving. While the 'leader' is observing, he proposes the next actions, dictates word by word or verbalizes the activity. We do not expect that a behavior like dictating exactly what to write might occur in balanced professional settings.

As a conclusion I state that it does not seem possible to extrapolate the results from student pairs to professionals in general. Comparability seems to depend on the aspect of the study and furthermore on choosing an adequate student pair. The quantitative analysis has shown that one possible filter could be the speech distribution and the speech contributions during the driving times of each programmer in order to select student pairs with more similarity to professional programmers. Self-estimated pair programming experience and quality of cooperation cannot be used as a filter.

7 Further Work

To determine the differences between professional and student pair programmers in greater detail, it could be helpful to study the process qualitatively. Furthermore, students in different

educational stages, like graduated students or students solving more complex problems in bigger software systems, should be compared with professional programmers. Another investigation could focus on the behavior of the professional sand whether or not it was better for the success of pair programming. If this is the case, the question emerges how I could train the students to be better pair programmers.

8 Acknowledgements

The author would like to thank the participating students and the participating company: disy Informationssysteme GmbH.

References

1. S. Bryant, P. Romero, and B. du Boulay. The Collaborative Nature of Pair Programming. *The 7th International Conference on Extreme Programming and Agile Processes in Software Engineering*, 2006.
2. Sallyann Bryant. Double trouble: Mixing qualitative and quantitative methods in the study of extreme programmers. In *VLHCC '04: Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing*, pages 55–61, Washington, DC, USA, 2004. IEEE Computer Society.
3. L. Cao and P. Xu. Activity patterns of pair programming. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Washington, DC, USA, 2005. IEEE Computer Society.
4. J. Chong and T. Hurlbutt. The Social Dynamics of Pair Programming. *Proceedings of the 29th International Conference on Software Engineering*, pages 354–363, 2007.
5. B. Curtis. By the Way, Did Anyone Study Any Real Programmers? *Empirical Studies of Programmers*, pages 256–262, 1986.
6. Brian Hanks, Charlie McDowell, David Draper, and Milovan Krnjajic. Program quality with pair programming in cs1. In *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 176–180, New York, NY, USA, 2004. ACM Press.
7. Andreas Hfer. Video analysis of pair programming. In *APOS '08: Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral*, pages 37–41, New York, NY, USA, 2008. ACM.
8. Kim Man Lui and Keith C.C. Chan. When does a pair outperform two individuals? In *Extreme Programming and Agile Processes in Software Engineering*, volume 2675 of *Lecture Notes in Computer Science*, pages 225–233. Springer, 2003.
9. Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. The effects of pair programming on performance in an introductory programming course. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 38–42. ACM Press, 2002.
10. Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, and Suzanne Balik. Improving the cs1 experience with pair programming. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pages 359–362, New York, NY, USA, 2003. ACM Press.
11. Nachiappan Nagappan, Laurie A. Williams, Eric Wiebe, Carol Miller, Suzanne Balik, Miriam Ferzli, and Julie Petlick. Pair learning: With an eye toward future success. In *XP/Agile Universe*, volume 2753 of *Lecture Notes in Computer Science*, pages 185–198. Springer, 2003.
12. P. Romero S. Bryant and B. du Boulay. Pair programming and the mysterious role of the navigator. *International Journal of Human-Computer Studies*, in press.
13. TechSmith Corporation. Camtasia studio 4.0.1. http://www.techsmith.com.
14. Laurie Williams and Robert R. Kessler. Experimenting with industry's "pair-programming" model in the computer science classroom. *Journal of Software Engineering Education*, december 2000.
15. Laurie Williams, Robert R. Kessler, Ward Cunningham, and Ron Jeffries. Strengthening the case for pair programming. *IEEE Software*, 17(4):19–25, 2000.
16. Laurie Williams and Richard L. Upchurch. In support of student pair-programming. In *SIGCSE '01: Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*, pages 327–331, New York, NY, USA, 2001. ACM Press.
17. S. Xu and V. Rajlich. Dialog-based protocol: an empirical research method for cognitive activities in software engineering. In *International Symposium on Empirical Software Engineering*, pages 383–392, 2005.