Evaluating Tools that Support Pair Programming in a Distributed Engineering Environment

Dietmar Winkler   Stefan Biffl   Andreas Kaltenbach

Institute of Software Technology and Interactive Systems
Vienna University of Technology, Austria
Favoritenstrasse 9-11/188, 1040 Vienna, Austria
{Dietmar.Winkler, Stefan.Biffl}@tuwien.ac.at

The construction and improvement of high-quality products in a global software development environment requires (a) flexible practices for collaboration and (b) tools that support these practices in a distributed software development team. Pair Programming (PP), a well-known agile practice has been reported to improve software productivity and software quality in co-located environments. However, in a distributed environment PP needs stronger tool support to address new challenges like communication, distributed collaboration, and data exchange. This paper introduces a systematic tool evaluation approach for distributed pair programming (DPP) and reports on an initial tool survey of open source tools. Major findings were that (a) DPP was not fully supported by any tool under investigation and (b) some tools are limited to selective and individual aspects of DPP requirements. The results of this tool evaluation can help project managers in selecting adequate tools and tool developers in providing high-value features for better support of DPP as basis for improving the quality of distributed engineering projects.

Keywords: Global Software Development, Tool Support, Tool Evaluation Survey, Distributed Pair Programming

1. INTRODUCTION

The increasing need for flexible software development, e.g., response to frequently changing requirements and faster delivery of software components at a high level of software quality foster the application of agile software processes, e.g., Scrum [3] and eXtreme Programming [2]. The construction of high-quality software products requires appropriate practices for product construction. Pair Programming (PP), a well-established agile practice, is applicable to Scrum and eXtreme Programming and enables the co-located development of high-quality software components [21][22]. A traditional PP team consists of two persons working concurrently on the same artifact, sharing a common working environment, i.e., one keyboard, one mouse, and one monitor. Empirical studies [13][16][21] have demonstrated, that PP can increase productivity, quality, and job satisfaction.

The trend towards distributed software development in a global environment aims at decreasing development cost, development duration, and travelling effort. This approach builds on geographically distributed software engineering teams. Typically, distributed software development focuses on the establishment of on-site development centres and client-vendor relationships on business and management levels [18] with defined interfaces. However, there is still a shortage regarding cooperation on technical level, i.e., code construction, as geographic distances and associated attributes of culture, time zones, and language, affect how humans interact with each other [15] and must be supported by processes, methods and tools to enable efficient distributed software development approaches on a technical level.

As traditional PP fosters co-located development distributed pair programming (DPP) aims at increasing product quality and productivity over distances. However, DPP raises a new set of issues regarding communication and collaboration [12] and therefore requires powerful tools to support communication, collaboration, and data exchange to bridge geographical and temporal distances. In common industry environments a wide range of tools exists to support collaboration. Thus, important questions include (a) deriving most relevant tool requirements in a specific application context and (b) evaluating candidate tools regarding these requirements efficiently.

This paper presents (a) a systematic process to derive requirements for tool evaluation and demonstrates the application of this process in an initial tool survey. Additionally, we present first results of an evaluation process for selecting DPP tools.

The remainder of this paper is structured as follows: Section 2 provides related work on PP, DPP, and tool...
evaluation. Section 3 introduces the research questions and section 4 provides an overview of an extended tool evaluation process. We provide results on the evaluation of open source DPP tools in section 5 and limitations in section 6. Finally, section 7 concludes and suggests future work.

2. RELATED WORK

This section summarizes related work on pair programming (PP) and distributed pair programming (DPP) and derives the need for efficient tool support.

2.1 Pair Programming

Agile processes include a wide range of engineering practices such as Test-Driven Development [9] and PP [22]. Traditional PP is a co-located agile practice embedded within an agile software process, e.g., SCRUM [3] and eXtreme programming [2]. Following the definition given by Williams and Kessler [22] the term PP describes the interaction and collaboration of two software developers sitting side by side in front of the same computer screen and collaboratively working on the same design, algorithms, code, or test. One of the two developers (Driver) controls the keyboard and mouse and actively works on the current task, i.e., implementing the design or code. The second member of the PP team (Observer or Navigator) supports the driver by providing useful information, identifying defects (i.e., performing continuous reviews), and keeping the strategic overview on the next steps. Effective PP is a highly collaborative activity involving continual conversation and gesturing [22]. The two roles, Driver and Navigator, are exchanged arbitrarily between the two programmers building the PP team. Both partners are completely emancipated and actively participate in the development all the time. No line of code is written without the other partner’s presence and agreement. They discuss various possibilities of implementing the same functionality and both partners introduce their individual knowledge and experience.

Previous research has indicated that PP results in better quality than individual programming in situations where the pair programming team is physically colocated as well as in such situations where this is not the case [14]. Furthermore, the cost and benefit ratio of PP in colocated teams has been thoroughly researched [16][21]. The outcome of this research strengthens the case for PP, also suggesting its appliance in distributed development teams. Williams et al. identified a set of synergy effects of PP in contrast to solo programming [21]:

- **Pair Pressure** indicates positive pressure due the presence of two developers; both participants contribute actively during the development process.
- **Continuous reviews** of the design and code, performed by the navigator that enables early defect detection and removal during pair development. Thus rework effort can decrease rapidly.
- **The involvement of two brains** enables synergy effects based on the experience of the pair and can lead to better solutions.
- **Pair learning.** Continuous collaboration fosters knowledge transfer between two pair programming members.
- **Enhanced job satisfaction** because of intensive collaboration and common success factors.

Note that the term “pair programming” is not limited to pair implementation, but can also include other pair activities, like pair design and pair testing. Nevertheless, the positive effect of PP might be a promising approach in a distributed environment.

2.2 Distributed Pair Programming

Following the definition by Baheti et al. distributed pair programming (DPP) refers to a geographically distributed collaboration of two engineers who concurrently collaborate on a given implementation task [1]. In contrast to traditional co-located PP, specific tools are required to support collaboration and communication of PP members to gain benefits from PP application. A DPP team can be compared to a virtual team, i.e., a 2-person-group of collaborating people with focus on a common goal across time, distance, culture, and organizational borders [10]. Previous research in the area of distributed collaboration [8] and DPP [12] identified some basic requirements for DPP tool support. Analyses on DPP application based on [7] and [12] produce a set of basic requirements:

- **Workspace Control and Awareness** refers to the knowledge of participating users, i.e., who is participating, which artifact is affected by the collaboration, and what has to be done next, i.e., the understanding of another person’s interaction within the shared space [11].
- **Screen sharing** enables a similar view on the same artifact by transferring the content of the screen to all participating parties.
- **Floor control** refers to alternating control to change the current piece of work, i.e., the driver can provide new information and the observer can observe these activities. A well-defined handshake mechanism must be implemented to change role assignments.
- **Gesturing** is a very important mechanism in the collaboration of people, because it enables the visualization of certain aspects on the screen. For instance the observer can focus on an identified defect on the screen by pointing to the defect with a second mouse pointer. Tang et al. identified gesturing as a success-critical element of group work [20]. In DPP, Cox and Greenberg [7] as well as Hanks [12] identified gesturing and deictic references as integral requirements of co-located PP.
• Availability of various communication channels. As PP includes intensive collaboration and communication between PP-team members, information exchange via text-messages, voice, or video is required for effective collaboration. Canfora et al. [6] found that separated PP team members who are limited to text-based chat tend to stop collaborating and start to work individually. Observations made by Hanks [12], Baheti et al. [1] and Schümmer et al. [19] identified that communication via video is not a mandatory element to support DPP; message exchange and voice communication can be sufficient for efficient DPP application.

• Platform independence, usability, and tool documentation can support various operation systems settings, easy installation and intuitive application.

Therefore, pre-conditions for the successful application of DPP in a distributed environment include (a) simultaneous collaboration regarding the same artifact within the same working environment and (b) efficient communication approaches. The selection of the most valuable tool for distributed collaboration support is a major issue in the context of DPP practice.

2.3 Tool Evaluation Process

Efficient tool selection and evaluation processes are key issues in software engineering to increase development efficiency. As the distributed application of methods relies on efficient tools for collaboration and simultaneous work, appropriate approaches are required to address these additional issues. Based on a wide range of available solutions, a systematic approach for tool evaluation is required. Poston et al. [17] provide a general purpose workflow for tool evaluation with respect to testing tools, which seems appropriate for DPP tool selection and evaluation.

![Figure 1: Tool selection and evaluation process based on Poston et al. [17].](image)

Poston et al. identified four major steps in a process for selecting and evaluating testing tools (see Figure 1): (1) Analysis and classification of requirements and expected tool properties; (2) Definition of tool selection criteria including prioritization; (3) Search for tools and classification of candidate tools; (4) Evaluation of candidate tools with respect to a given evaluation framework. The evaluation process can be supported by a spreadsheet, including required tool properties and an evaluation schema for assessing the degree of requirements and property coverage for every tool. Nevertheless, individual requirements have to be addressed and figured out to focus on the individual application domain, in our case DPP applications.

3. RESEARCH ISSUES

DPP requires effective tool support to address the need and the basic requirements of the PP team for effective and traceable collaboration. From these general needs we derive the following research issues:

1. Providing a systematic evaluation approach for DPP tool assessment. A comprehensive evaluation framework with respect to tools and important application requirements and tool properties is necessary to achieve comparable, traceable, and repeatable evaluation results. Based on the tool evaluation process for testing tools proposed by Poston et al. [17], we adapted the process approach to DPP supporting tools and applied this adapted evaluation process to a set of candidate DPP tools. Nevertheless, application requirements and tool properties are success-critical issues for DPP application. Thus, a second research issue focuses on the identification of basic DPP requirements.

2. Eliciting basic requirements and tool properties for DPP tool support. Based on the initial classification (see section 2.2) a more detailed view on requirements and tool properties is necessary to assess candidate tools and to better understand individual tool contributions. We applied a requirements elicitation workshop according to Boehm et al. [5] to identify detailed requirements and relevant properties for tool assessment.

4. TOOL EVALUATION APPROACH

The application of a systematic tool evaluation approach supports project and quality managers in selecting best-practice tools for DPP application and can support tool vendors in (a) better understanding the needs of the users, (b) assessing the tool with respect to current and upcoming user requirements, and (c) identifying improvement capabilities to increase tools performance and tool applicability. Based on the general purpose tool evaluation approach proposed by Poston et al. [17], we identified a set of activities regarding the evaluation process to efficiently evaluate tools in context of DPP application. The adaptations include a classification and prioritization of requirements and a classification of tools. Classifying user requirements and tool categories can help to efficiently focus on a certain class of requirements (e.g., gesturing support) and tools (e.g., screen-sharing applications), independent of the implementation, e.g.,
The extended and refined tool evaluation process consists of five basic steps:

**Step 1: Analysis and classification of requirements and tool expected properties.** This step includes the identification of application-related requirements and expected tool properties. Typically, this initial step is based on previous analysis results (e.g., based on [7] and [12] regarding DPP requirements) and requirements brainstorming workshops, conducted by involved stakeholders (e.g., supported by EasyWinWin processes [5]). Prioritizing requirements according to their individual importance in the application domain can help in getting a more focused view during the evaluation. Additionally, classifying the requirements can help to identify groups of requirements with focus on individual and critical aspects of tools under investigation. See Appendix A for a summary of collected tool requirements and properties.

**Step 2: Search and categorization of candidate tools.** Classified requirements and tool properties are the baseline for searching for candidate tools. A preliminary selection of candidate tools can be based on tool descriptions. Depending on the identified candidate tools, a categorization of tools can help identifying families of tools for specific purposes. In DPP we identified two major categories of tools: (a) screen sharing applications and (b) distributed collaborative work applications. Nevertheless, sub-categories can be introduced to classify tools more precisely.

**Step 3: Evaluation framework definition.** The application of a spreadsheet model is a typical strategy for evaluating tools with respect to requirements and candidate tools. Every candidate tool is evaluated with respect to every prioritized requirement. The summary of the evaluation results can be seen as tool matrix.

**Step 4: Scenario Brainstorming.** Scenarios can help focusing on typical activities and conditions to better identify benefits and weaknesses of the object under investigation, i.e., a tool for DPP support. This evaluation step includes the identification of common scenarios in context of application requirements and expected tool properties. Note that scenario brainstorming processes can also be supported by EasyWinWin [5].

**Step 5: Tool evaluation and assessment of candidate tools according to the evaluation framework by applying identified scenarios.** Note that the evaluation also includes aggregation to requirements classes and tool categories, and the interpretation and discussion of evaluation results.

The systematic application of this tool evaluation approach can enable comparable, traceable and repeatable results because the evaluation is based on common classified requirements and properties and a defined set of application scenarios.

### 5. DISTRIBUTED PAIR PROGRAMMING TOOL EVALUATION

We applied the proposed evaluation process by conducting an initial tool survey for DPP supporting tools. The goal of the survey was to (a) setup an evaluation framework for DPP tool evaluation, (b) elicit basic DPP tool requirements and properties and scenarios, and (c) evaluate selected tools according to the defined evaluation framework.

#### 5.1 Survey Design and Arrangement

The tool survey was conducted by 8 software engineering master students supported by a supervisor. All students had professional software engineering experience from an industry context for at least 3 years. Nevertheless, they had a tutorial (2h) on pair programming and empirical software engineering prior to the survey. All participants contributed in brainstorming activities, i.e., requirements and tool property elicitation (section 5.2) and scenario brainstorming (section 5.5). Regarding the tool evaluation tasks the participants were grouped randomly to pairs who evaluated randomly assigned tools with respect to the evaluation framework and the scenarios.

Requirements and expected tool properties were collected by all participants in an EasyWinWin workshop [5] in two iterations, i.e., initial brainstorming and requirements categorization and prioritization (section 5.2). The participants were supported by basic requirements categories (see section 2.2) to stimulate brainstorming activities. Scenarios for DPP application were collected during a second EasyWinWin workshop (section 5.5). We conducted a systematic tool search for candidate tools (section 5.3) and decided to include them in the survey based on availability, licensing issues, and product descriptions. The study followed five steps as described in section 4. We used a spread-
sheet to setup the evaluation framework and to capture the results of the tool evaluation (section 5.6). Finally, we will discuss relevant limitations and threats to validity in section 6.

| General tool Properties (8) | Priority |
|-----------------------------|----------|
| General Requirements (2)    | Priority |
| Support of category supported features (e.g., Screen Sharing, Collaborative Work Support) | Critical (C) |
| Support of Workspace Awareness | Critical (C) |
| Floor Control (7)           | Priority |
| Support of Floor Control    | Critical (C) |
| Support of Role Changes     | Critical (C) |
| Role Assignment information | Medium (M) |
| Gesturing (4)               | Priority |
| Second Pointer for the Navigator | Critical (C) |
| Support of Highlighting     | Low (L) |
| Communication (5)           | Priority |
| Voice Channels              | High (H) |
| Textual Chat                | Medium (M) |
| Video channel               | Low (L) |
| Platform Independence (3)   | Priority |
| Usability (10)              | Priority |

![Figure 3](image_url) Snapshot of the prioritized requirements list for DPP application.

5.2 Identification and Classification of Requirements and Tool Properties

Capturing and prioritization requirements and tool properties represents the first step in the evaluation process with respect to framework construction. Based on previous work [6][12] in the area of DPP application, we identified a set of basic requirements and expected tool property classes (see section 2.2 for details). This classification was used to stimulate the brainstorming process during the capturing process. To identify individual requirements within the relevant classes, we conducted a requirements brainstorming and a prioritization workshop and achieved an overall number of 39 individual requirements (see Appendix A1), which should be addressed for specific tool supporting DPP application.

Figure 4 presents a snapshot of requirement categories and samples of individual requirements. Note that the numbers beside the classified properties indicate the number of involved detailed requirements. The prioritization step of individual requirements was conducted in a second workshop to strengthen the contribution of property according to the DPP application. Every priority was assigned to a specific weight representing the value of each requirement partly based on previous publications [1][6][20]. We used a scale of 4 priority levels: critical requirement (C, weighted with 10) are success-critical issues for DPP application, high important requirements (H, weighted with 5) have a strongly positive impact on DPP application success, medium importance (M, weighted with 2.5) have some positive aspect, assumed to be not success-critical, and low important requirements (L, weighted 1) can be considered as nice to have features, i.e., can help DPP application but do not hinder if the requirement is not supported by the candidate tool.

5.3 Identification and Classification of Candidate Tools

The second step focuses on the identification of candidate tools for DPP support. Thus we conducted a systematic tool search via Internet with focus on teamwork, collaboration and distributed development. Additionally we analysed previous publications, e.g., [1] and [15] to identify additional and missing tools. Based on this research we grouped the results list of tools according to similar behaviour of the tools and identified two groups of basic candidate tools.

(a) Screen sharing applications with and without interaction. Screen sharing applications without interaction focus on helpdesks and system administrator tasks, e.g., the “client” (observer) shares the screen and the administrator conducts some defined service tasks (comparable to the driver view). Screen sharing application with interaction typically support whiteboards and chat as required by video conferences.

![Figure 4](image_url) Classified and Selected Candidate Tools for DPP application.
(b) **Collaboration-aware applications**, i.e., distributed editors and collaboration integration within IDEs. Note that the evaluation also includes tools, especially designed for DPP application. The basic idea of DPP has led to the extension of various tools to support DPP, e.g., Hanks VNC extension [12], PEP (Eclipse Plugin for Distributed Java Pair Programming), and XPairtise (Eclipse Plugin for DPP). Figure 4 presents a selection of candidate tools for DPP application.

Because of the wide range of tools available, we focus on a subset of candidate tools, promising to be most valuable for DPP application (highlighted in Figure 4). Note that we focus on 2 pure screen-sharing applications and 10 collaborative work supporting tools in detail. The selection of these tools was based on tool availability, licensing issues (we focus on OSS tools), and documentation.

### 5.4 Evaluation Framework Definition

After requirements and tool property identification and classification in step 1 and an initial survey regarding candidate tools for DPP support in step 2, the third step includes the preparation of the evaluation framework. We applied a tool matrix for evaluation purposes. Figure 5 presents a schematic overview on the tool evaluation matrix including all classified requirements (derived from 5.2) including prioritized individual requirement on the y-axis and selected candidate tools on the x-axis.

![Figure 5: Schematic overview of the evaluation framework.](image)

The basic idea of the evaluation of individual requirements with respect to selected tools is an estimation of the degree of requirements coverage during a sample application, i.e., during scenario application. This sample application includes typical tasks in context of tool usage including installation processes. To identify the applicability of the tool in context of DPP, we used a set of typical scenarios representing typical tasks from an agile software project and “simulated” DPP application for these tasks. Afterwards, the participants (each pair) gave a rating on the fulfillment of the tool property with respect individual requirements. We used two types of ratings: (a) range from 0 (not supported) to 5 (requirement is fully supported by the tool) if a stepwise assessment will be reasonable and (b) a binary rating (0 not supported / 1 supported) if no in-between ratings are available (e.g., tool support for an individual operating system). This individual rating was weighted according to requirements priorities. Note that this systematic approach enables comparability, traceability and repeatability, because the evaluation is based on similar and prioritized requirements.

### 5.5 Scenario Brainstorming

Scenarios can help to observe activities under typical conditions to better identify benefits and weaknesses of the object under investigation. Therefore, we applied scenarios derived from application requirements, which are typically relevant for a tool application in a DPP setting. Typical scenarios were identified during a scenario brainstorming workshop, as a common approach in other disciplines like scenario brainstorming processes in architecture evaluation [4]. To achieve a common understanding of the scenarios and enable comparable evaluation results we provided a set of guidelines for scenario application purposes during the evaluation process. In general, we identified 2 scenario categories and an overall number of 6 relevant scenarios:

(a) **General scenarios** focus (a) on the installation process of the tool and (b) the general handling of the tool without focusing on DPP application.

(b) **Specific DPP scenarios** focus on individual tasks, common in a DPP setting: (a) initialization of a DPP session, (b) role support and change of role (floor control), (c) session management (storing/restoring sessions), and (d) tool application to solve a small implementation task.

### 5.6 Evaluation of Candidate Tools

Focus of the evaluation was to identify benefits and weaknesses of the applied tool with respect to individual requirements in DPP. Real distributed pairs solved the defined scenarios by applying candidate tools. The evaluation itself consists of subjective estimation values of tool usefulness provided by the real pairs. The results were checked by the supervisors for plausibility and correctness. Figure 6 presents a snapshot of selected candidate tools for DPP application including the pair ratings after applying the scenarios. Note that the figure includes representatives of (a) pure screen sharing applications (i.e., VNC and Netmeeting) and (b) collaborative work tools designed to support DPP activities (i.e., Copper, PEP).

![Figure 6: Snapshot of selected candidate tools for DPP application.](image)

For instance, there are notable benefits regarding **floor control and role changes** for DPP designed tools. Nevertheless, Netmeeting (a screen sharing application) had similar ratings compared to PEP (a DPP designed tool). An interesting fact is that **highlighting** is fully supported by VNC but there is limited support by Copper and PEP.
Because of various benefits and weaknesses of the candidate tools regarding individual requirements it is hard to identify a best-practice tool (note that we investigated an overall number of 39 requirements and tool properties). To get a specific view on every requirements category, we aggregated the results of the individual requirements per category by summing up the individual weighted ratings and classified the overall rating per category in three thirds: the first third (i.e., 0-33%, red marked) was classified as no/low support, the middle third (i.e., 33-66%, orange marked) include a medium support for DPP application and the upper third (i.e., 66-100%, green marked) was found to support DPP requirements quite well. Note that the introduced threshold values can be modified to achieve a more detailed classification of candidate tools. Figure 7 presents the results of the selected candidate tools with respect to the aggregated assessment result according to requirements categories.

### Figure 6: Snapshot of the Evaluation Matrix after Pair Assessment.

| General Tool Properties (8) | Rate | Weighted | Rate | Weighted | Rate | Weighted | Rate | Weighted | Rate | Weighted |
|-----------------------------|------|----------|------|----------|------|----------|------|----------|------|----------|
| General Requirements (2)    |      |          |      |          |      |          |      |          |      |          |
| Support of category supported features Critical (C) | 4 | 40 | 4 | 40 | 4 | 40 | 3 | 30 |
| Support of Workspace Awareness Critical (C) | 5 | 25 | 4 | 40 | 4 | 40 | 2 | 20 |
| Floor Control (7)           |      |          |      |          |      |          |      |          |      |          |
| Support of Floor Control Critical (C) | 0 | 0 | 3 | 30 | 5 | 50 | 3 | 30 |
| Support of Role Changes Critical (C) | 0 | 0 | 2 | 20 | 5 | 50 | 4 | 40 |
| Role Assignment information Medium (M) | 0 | 0 | 2 | 2 | 5 | 12,5 | 4 | 10 |
| Gesturing (4)               |      |          |      |          |      |          |      |          |      |          |
| Second Pointer for the Navigator Critical (C) | 1 | 1 | 1 | 10 | 0 | 0 | 0 | 0 |
| Support of Highlighting Low (L) | 5 | 25 | 2 | 2 | 2 | 12,5 | 1 | 1 |
| Kommunikation (5)           |      |          |      |          |      |          |      |          |      |          |
| Voice Channels High (H)     | 0 | 0 | 5 | 25 | 1 | 5 | 0 | 0 |
| Textual Chat Medium (M)     | 0 | 0 | 4 | 10 | 3 | 7,5 | 5 | 12,5 |
| Video channel Low (L)       | 0 | 0 | 5 | 5 | 0 | 0 | 0 | 0 |
| Platform (3)                |      |          |      |          |      |          |      |          |      |          |
| Usability (10)              |      |          |      |          |      |          |      |          |      |          |

Figure 7: Snapshot of the Aggregated Evaluation Matrix per Requirements Category.

#### 5.7 Results of the Tool Evaluation

Figure 8 presents the summarized results of the tool evaluation for all evaluated candidate tools. Note that we calculated the share of the assessment results per requirements class with respect to the maximum value per class. Additionally we sorted the overall rating according to the assessment value. The results show that basic requirements (i.e., supporting screen sharing and/or collaborative work), platform independence, and usability were supported quite well. Additionally, we observed some tool-related improvement issues, e.g., improvement during installation processes. Core features for DPP application, like gesturing, floor control, and communication support were only supported for a view tools under investigation. Thus, there is potential for tool improvement to address basic requirements for DPP tool application. Note that Xpairtise and Copper have been specifically designed for DPP application. It is no surprise that both tools are top-ranked during the evaluation. A
The surprising fact was that a pure screen-sharing application, i.e., Net-Meeting and a collaborative editor, i.e., TalkAndWrite seem to support specific tasks of the DPP quite well.

| Tool           | Screen-Sharing | Collaboration | Basics | Gesturing | Floor Control | Communication | Platform | Usability | Tool | Rate | Weighted | Weighted [%] |
|----------------|----------------|---------------|--------|-----------|---------------|---------------|----------|-----------|------|------|----------|--------------|
| Xpairtise      | X              |               | 100%   | 22%       | 82%           | 38%           | 100%     | 83%       | 56%  | 100 | 471      | 72%          |
| Copper         | X              |               | 80%    | 11%       | 89%           | 29%           | 100%     | 83%       | 31%  | 85  | 442      | 68%          |
| TalkAndWrite   | X              |               | 90%    | 85%       | 0%            | 83%           | 42%      | 73%       | 75%  | 91  | 396      | 61%          |
| Net-Meeting    | X              |               | 80%    | 30%       | 42%           | 83%           | 42%      | 69%       | 69%  | 87  | 390      | 60%          |
| PEP            | X              |               | 50%    | 5%        | 66%           | 29%           | 100%     | 73%       | 50%  | 79  | 351      | 54%          |
| DocSync        | X              |               | 80%    | 0%        | 63%           | 24%           | 100%     | 60%       | 56%  | 72  | 342      | 52%          |
| VNC            | X              |               | 90%    | 41%       | 11%           | 0%            | 83%      | 76%       | 44%  | 69  | 328      | 50%          |
| Gobby          | X              |               | 90%    | 5%        | 0%            | 21%           | 100%     | 82%       | 50%  | 61  | 304      | 47%          |
| GrewpEdit      | X              |               | 70%    | 16%       | 0%            | 25%           | 100%     | 78%       | 25%  | 56  | 284      | 44%          |
| MoonEdit       | X              |               | 60%    | 16%       | 0%            | 0%            | 100%     | 86%       | 56%  | 58  | 262      | 43%          |
| NetBeans Coll. | X              |               | 60%    | 0%        | 8%            | 29%           | 100%     | 63%       | 81%  | 68  | 249      | 38%          |
| ACE            | X              |               | 50%    | 22%       | 8%            | 8%            | 83%      | 71%       | 75%  | 57  | 250      | 38%          |

Maximum reachable Assessment values: 159 652

Figure 8: Results of The Tool Evaluation for Candidate DPP Tools.

6. LIMITATIONS AND VALIDITY CONSIDERATIONS

In order to increase internal and external validity we considered a set of threats and implemented appropriate countermeasures to address them:

- **Tool evaluation process.** We based the evaluation approach for DPP tool evaluation on a general-purpose and well-established process for tool selection and evaluation [17].

- **Requirements and scenario capturing.** We captured basic requirements in a requirements elicitation workshop [5] and used previous research results [12] to identify relevant requirements for DPP tools. The selected scenarios, derived from a scenario brainstorming workshop, represent important and typical real-world scenarios. Thus, interpretations of the results are limited to these scenarios. Nevertheless, the selected scenarios represent critical requirements of DPP.

- **Tool selection** is based on availability and licensing issues of the tools. Thus, we did not capture commercial tools and focused on a selection of most promising candidate open source software (OSS) tools to support DPP.

- **Participants.** The study was conducted by graduate students of computer science using a small implementation task (in Java) to assess DPP capabilities of the selected tools. The participants had a tutorial regarding software engineering and empirical software engineering. Furthermore, all participants work in an industry setting with profession background. Nevertheless, none of them had prior experiences in pair programming. This setting might not be representative for an industrial setting. Nevertheless, this will not be a strong limitation to evaluate the tool features with respect to DPP requirements as the scenarios can be extended in an industry setting.

7. CONCLUSION AND FURTHER WORK

The ongoing trend towards globally distributed software development and the need for flexible and fast delivery of software products in short iterations leads to the application of flexible and agile software processes, supported by agile best-practices. Traditional PP is a well-established approach in agile development but is limited to co-located software development. Distributed pair programming (DPP) is a candidate approach for globally distributed and flexible teams, if supported with sufficiently powerful tools. Nevertheless, a systematic evaluation of tools is an important approach for best-practice tool selection in a defined project environment. In this paper we addressed the following research issues.

- **Basic requirements for DPP tool support.** We identified seven basic requirements classes (i.e., basic properties, gesturing support, floor control, communication, platform independence, usability, and tool related properties) and an overall number of 39 detailed attributes to assess individual tool performance for DPP application (see Appendix A1 for details).

- **Candidate DPP tools.** During our research on DPP supporting tools we observed two major tool categories (a) screen-sharing applications and (b) collaboration-support tools. Additionally, various extensions of existing tools for DPP support were developed recently. The
evaluation of tools. Efficient tool evaluation processes are required to support efficient tool selection. Based on Poston et al. [17] we introduced an extended tool evaluation process in 5 steps and demonstrated an initial tool survey with respect to DPP tool support. The application of a systematic evaluation process approach enables comparable, traceable and repeatable results.

Major findings of the initial tool survey were (a) that there is no tool available that comprehensively supports DPP applications and (b) several tools support individual aspects of DPP requirements well, e.g., gesturing, floor control, and communication approaches. While tools designed for DPP support achieved somewhat better results in specific areas, there is still much work to do to provide a comprehensive tool support for DPP tasks.

Future Work. Based on the initial results of the tool survey we identified a set of aspects, relevant for future work: (a) improvement and evaluation of the proposed evaluation process based on the initial findings to include additional requirements and to refine the rating schemes in industry settings; (b) extending the number of tools for DPP application to address new, improved (modified), and commercial tools towards a comprehensive overview on the rating of current DPP tools; (c) Pilot application of selected tools in real-world projects including measurement of performance and quality attributes. Scenarios used in this tool survey cover basic DPP tasks. Real-world projects can provide a deeper insight in scenarios from industry and can help improving the evaluation framework.

REFERENCES

[1] Baheti P., Williams L., Gehringer E., Stotts D., Smith J.: Distributed Pair Programming: Empirical Studies and Supporting Environments. University of North Carolina at Chapel Hill, Department of Computer Science, TR02-010, 2002.

[2] Beck K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, 2000.

[3] Beedle M., Schwaber K.: Agile Software Development with SCRUM, Prentice Hall, 2008.

[4] Biffi S., Ali Babar M., Winkler D.: Impact of Experience and Team Size on the Quality of Scenarios for Architecture Evaluation, Conference on Evaluation and Assessment in Software Engineering (EASE), Bari, Italy, 26.-27.06.2008.

[5] Boehm B., Grünbacher P., and Briggs R.: Easy-WinWin: A Groupware-Supported Methodology for Requirements Negotiation, Int. Conf. on SE, 2001.

[6] Canfora G., Cimitile A., Vissagio C.A.: Lessons learned about distributed pair programming: what the knowledge needs to address, Proc of WETICE03, pp.314-319, 2003.

[7] Cox D., Greenberg S.: Supporting collaborative interpretation in distributed groupware. In CSCW ‘00: Proceedings of the 2000 ACM conference on Computer supported cooperative work, 2000.

[8] Damian D., Lanubile F., Oppenheimer H.L.: Addressing the Challenges of Software Industry Globalization, Workshop on Global Software Development. ICSE, pp793-794, 2003.

[9] Damm L.-O., Lundberg L.: Quality Impact of Introducing Component-Level Test Automation and Test-Driven Development, Proc. EuroSPI, 2007.

[10] George B., Mansour Y.M.: A Multidisciplinary Virtual Team, Systemics, Cybernetics and Inf., 2002.

[11] Gutwin C., Greenberg S.: The effects of workspace awareness support on the usability of real-time distributed groupware. In ACM Transactions on Computer-Human Interaction, 6(3), 1999.

[12] Hanks B.F.: Empirical Studies of Distributed Pair Programming. Dissertation in Computer Science, University of California Santa Cruz, December 2005.

[13] Müller M.M.: Do Programmer Pairs make different Mistakes than Solo Programmers? Proc of 10th Int. Conf on Evaluation and Assessment in Software Engineering (EASE), Keele Univ, 2006.

[14] Nawrocki J.R., Jasinski M., Olek L., Lange B.: Pair Programming vs. Side-by-Side Programming, Proc of 10th Int. Conf on Evaluation and Assessment in SE (EASE), Keele Univ, 2006.

[15] Olson G.M, Olson J.S.: Distance Matters, Human-Computer Interaction, vol 15, pp139-179, 2000.

[16] Padberg F., Müller M.M.: Analyzing the cost and benefit of pair programming, Proc of 9th Metrics Symp, pp.166-177, 2003.

[17] Poston R.M., Sexton M.P.: Evaluating and selecting testing tools, IEEE Software (9/3), 1992.

[18] Prikladnicki R., Damian D., Audy J.L.N.: Patterns of Evolution in the Practice of Distributed software Development: Quantitative Results form a Systematic Review, Proc of 12th Int. Conf on Evaluation and Assessment in SE (EASE), Bari, 2008.

[19] Schümmer T., Schümmer J: Support for distributed teams in extreme programming. In G. Succi and M. Marchesi, editors, Extreme Programming Examined, pages 355-378, Addison-Wesley, 2001.

[20] Tang J.C.: Findings from observational studies of collaborative work. In International Journal of Man Machine Studies, 34(2), pages 143–160, 1991.

[21] Williams L., Kessler R.R., Cunningham W., Jeffries R.: Strengthening the case for pair programming, IEEE Software 17(4), pp19-25, 2000.

[22] Williams L., Kessler R.: All I Really Need to Know about Pair Programming I Learned in Kindergarten. In Communication of the ACM May 2000/Vol 43 No. 5, pages 108-114, May 2000.
APPENDIX

A1. Basic requirement of DPP tools

| General Tool Properties (8) | Priority | Weight |
|-----------------------------|----------|--------|
| Commercial tool             | Low (L)  | 1.0    |
| Open source tool            | Low (L)  | 1.0    |
| Special purpose tool        | Low (L)  | 1.0    |
| IDEs                        | Low (L)  | 1.0    |
| Viewer                      | Low (L)  | 1.0    |
| Configuration Management / Versioning | Low (L)  | 1.0    |
| Installation and Configuration | Low (L)  | 1.0    |
| User Manual                 | Low (L)  | 1.0    |

| General Requirements (2) | Priority | Weight |
|--------------------------|----------|--------|
| Support of category related features | Critical (C) | 10.0  |
| Support of workspace awareness | Critical (C) | 10.0  |

| Floor Control (7) | Priority | Weight |
|-------------------|----------|--------|
| Support of Floor Control | Critical (C) | 10.0  |
| Support of Role Changes | Critical (C) | 10.0  |
| Social Floor Control (Informal) | Medium (M) | 2.5   |
| Formal Floor control | Medium (M) | 2.5   |
| Easiness of Role Changes | Medium (M) | 2.5   |
| Support of "Who is in which role" | Medium (M) | 2.5   |
| Duration of "Who is in which role" | Low (L) | 1.0   |

| Gesturing (4) | Priority | Weight |
|---------------|----------|--------|
| Availability of a second pointer for the navigator | Critical (C) | 10.0  |
| Semantic adaptions of pointers | Medium (M) | 2.5   |
| Supression of Telepointer jitter | Low (L) | 1.0   |
| Support of highlighting | High (H) | 5.0   |

| Communication (5) | Priority | Weight |
|-------------------|----------|--------|
| Speech Channels   | High (H) | 5.0    |
| Textual Chat      | Medium (M) | 2.5   |
| White-Board       | Medium (M) | 2.5   |
| Video-Channel     | Low (L)  | 1.0    |
| Pair Finding Support | Low (L)  | 1.0   |

| Plattform Independence (3) | Priority | Weight |
|----------------------------|----------|--------|
| Windows                    | Medium (M) | 2.5   |
| Unix/Linux                 | Medium (M) | 2.5   |
| MacOS                      | Low (L)   | 1.0    |

| Usability (10) | Priority | Weight |
|----------------|----------|--------|
| Intuitiveness of tool usage | High (H) | 5.0    |
| Easiness of installation and configuration | Low (L) | 1.0   |
| Easiness of session start-up / end | Medium (M) | 2.5   |
| Stability | High (H) | 5.0   |
| Real-time behavior | Critical (C) | 10.0  |
| Easiness of collaboration | Critical (C) | 10.0  |
| Intuitiveness of User Interface | Medium (M) | 2.5   |
| Syntax Highlighting | Low (L) | 1.0   |
| Need for other tools | High (H) | 5.0   |
| Support of session management | Medium (M) | 2.5   |