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

dietmar.winkler@tuwien.ac.at

http://qse.ifs.tuwien.ac.at

Motivation

Challenges in modern software development practices:

- Delivery of high-quality software products within short iterations.
- Ability to respond to frequent and changing requirements.
- Selection of best-practice tools for project application.

Manage these challenges …

- Ongoing global software development (24h development).
- Application of agile and flexible software processes for project planning, monitoring and control.
- Application of established (agile) practices, e.g., pair programming.

→ Distributed pair programming

→ Tool support to enable distributed and collaborative software development.

Goals:

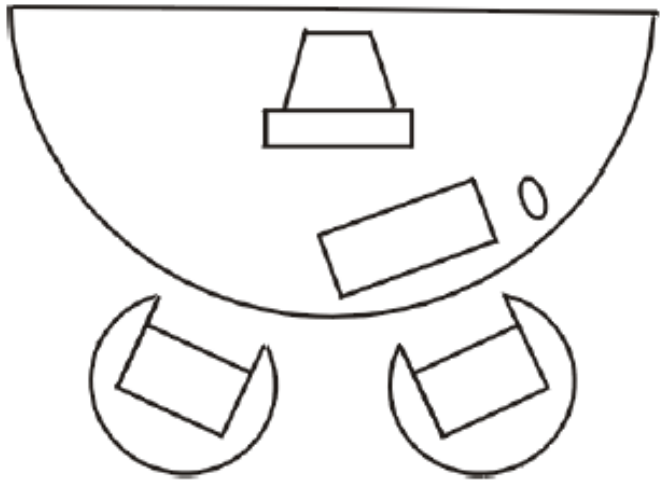
- Elicitation of basic requirements as foundation for distributed pair programming (DPP) application.
- Developing an evaluation framework for efficient toll evaluation.
- Application of the evaluation framework to identify tools for DPP support.

Traditional Pair Programming

- Pair programming is an established agile practice for efficient software development.
- Ability to support code construction, design, and test („pair activities“ and “pair tasks”).

- Basic idea of pair programming:
  - 2 developers (a pair) elaborate on a software artifact concurrently sharing a common working (co-located) environment (screen, keyboard, and mouse).
  - Clearly defined roles and change of role assignments:
    Driver (implementation) and Observer / Navigator (e.g., continuous reviews, provides)

- Benefits of pair programming (derived from various empirical studies):
  - Increased quality, effectiveness, and productivity.
  - Improved team communication.
  - Focus on a common artifact (“Pair Pressure”)
  - Pair learning.



- Is Pair Programming applicable in a distributed environment?

Distributed Pair Programming

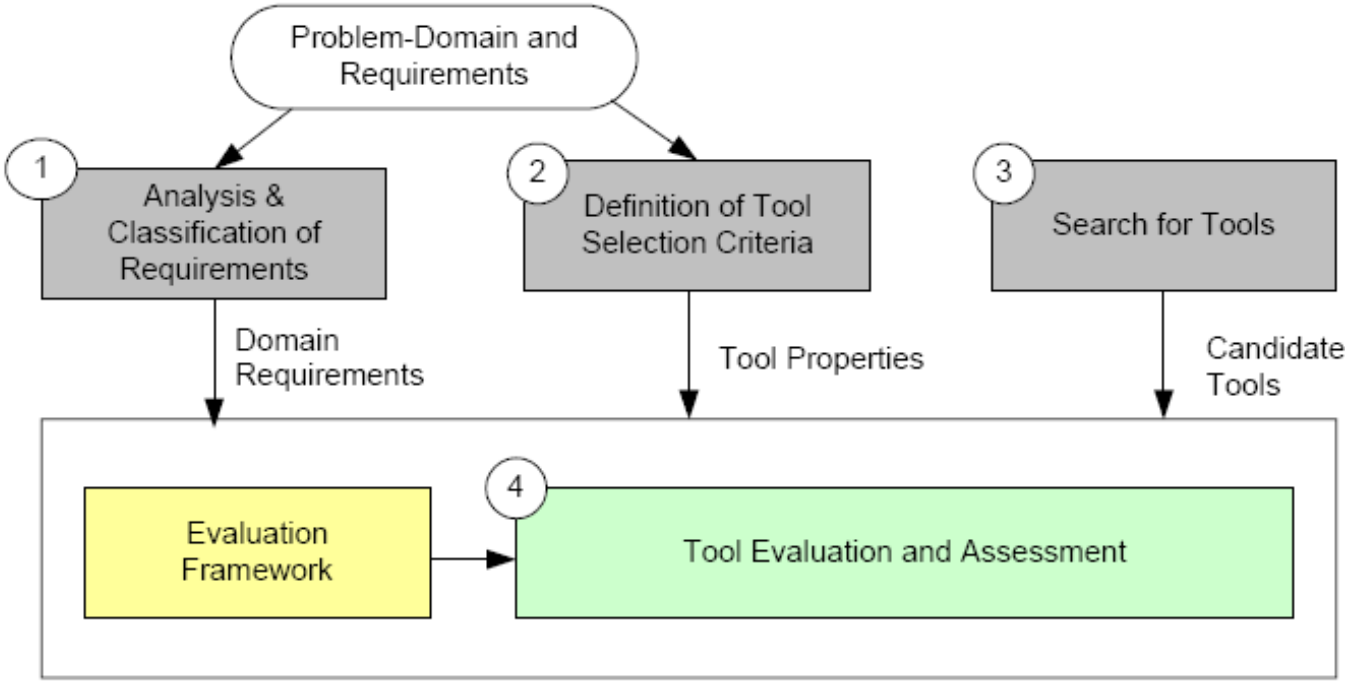
- Applying traditional pair programming requires a shared work space in a co-located environment.
- Basic idea of Distributed Pair Programming (DPP):
  - Pair programming and a shared workspace over distances.
  - Gaining benefits of traditional pair programming.
  - Continuous collaboration.

- Basic pre-conditions:
  - Efficient communication and collaboration mechanisms (e.g., screen sharing, communication channels, and gesturing).
  - Efficient data exchange approaches.
  - Workspace control and awareness (participants, artifacts, tasks).
  - Floor control (ability to change of roles and trace changes)
  - Tools that support collaboration and continuous interaction within a common working environment to bridge geographical and temporal distances.
- Challenge: Identification of a best-practice tool for DPP support.

Process for Tool Selection

Four basic steps for tool selection [Poston, 1992]:

1. Analysis and classification of user requirements and expected tool properties.
2. Elicitation and prioritization of selection criteria.
3. Classification of candidate tools.
4. Assessment of tools according to a pre-defined evaluation scheme.



[Poston, 1992]

Step 1: Requirements: Analysis of User Requirements

- Systematic research of existing analysis results regarding the application domain.
- Example: Distributed Pair Programming: Existing analysis results by Hanks, 2005 and Cox et al, 2000.

- Brainstorming of related stakeholders to capture and complete individual (tool) requirements.

- Samples of important basic requirements categories for DPP supporting tools:
  - Workspace control and awareness. Visibility of participants within the working environment; defined mouse and keyboard control.
  - Screen Sharing support.
  - Floor control. Transparent and traceable changes within an artifact by roles.
  - Gesturing. Ability to point to specific aspects of interest (e.g., defects) by using a second pointer device.
  - Efficient information exchange to support communication and collaboration, e.g., textual and/or voice chat, video conferencing.
  - Platform-independence, usability, tool documentation.

Step 2: Requirements: Classification and Prioritization

- Elicitation, classification and prioritization of collected requirements
- Requirements Elicitation workshop (EasyWinWin) according to Boehm et al, 2001.

- Snapshot of collected requirements.

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)
Plattform Independence (3)	Priority
Usability (10)	Priority

- Requirements prioritization

Prioritization	Weight
Critical	10
High importance	5
Medium importance	2,5
Low importance	1

- Critical: basic functions for DPP.
- High importance: Pre-conditions for efficient PP application.
- Medium importance: Requirements and attributes that can increase tool application.
- Low importance: Nice-to-have features.

Step 3: Candidate Tools: Search for Tools and Classification

Search for Candidate Tools:

- Identification of tools applicable for Computer Supported Collaborative Work:
  - Based on requirements categories.
  - Tools to support distributed collaborative work (independent of DPP).
  - Tools designed for DPP support.

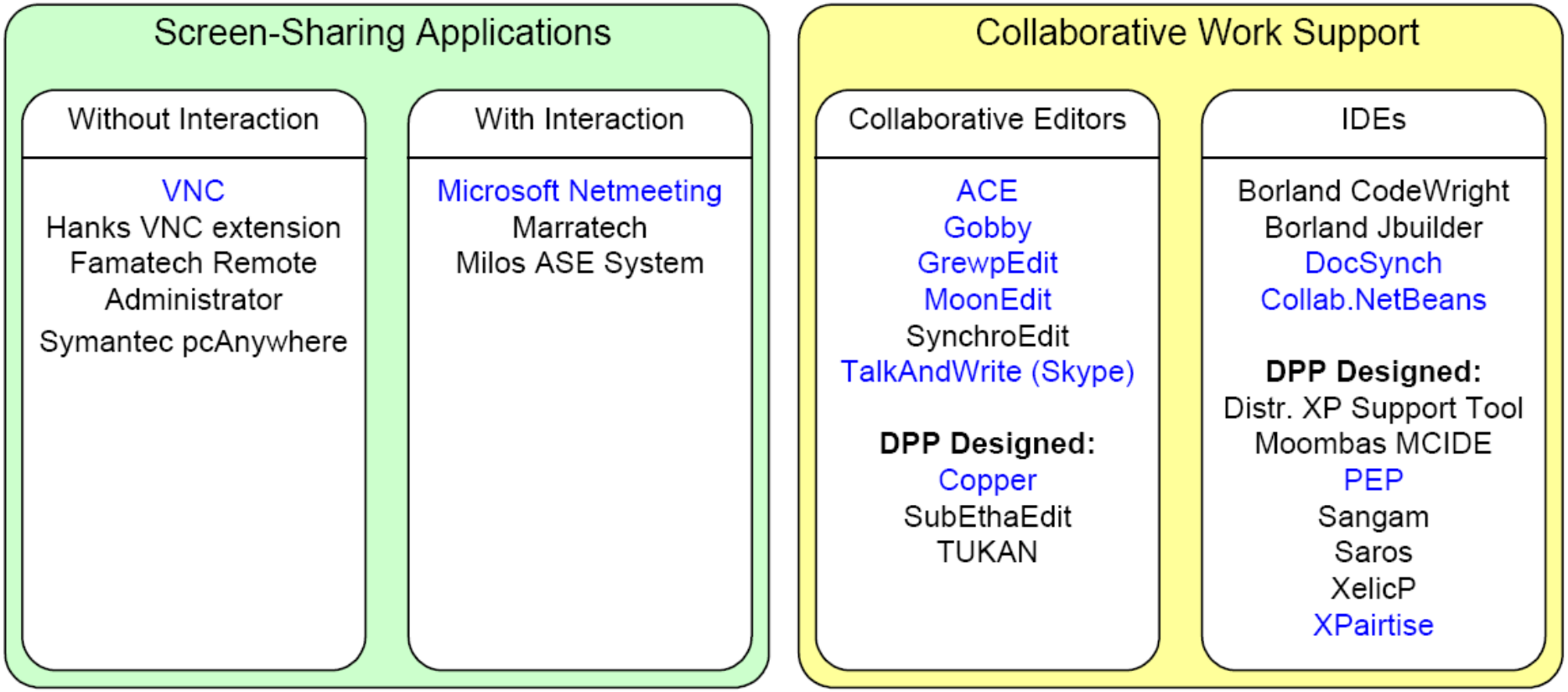
Basic Classification of Candidate Tools.

- Screen-Sharing Applications.
  - Screen-Sharing without Interaction: Exchanging screen content (e.g., VNC) typically used by system administrators.
  - Screen-Sharing with Interaction: Additional features like Whiteboards, Chat (e.g., MS Netmeeting). Typical used for video conferencing.
  - Tools for explicit DPP support

- Collaboration-aware applications.
  - Distributed editors
  - Integrated Development Environments (IDE).

Snapshot of Candidate Tools

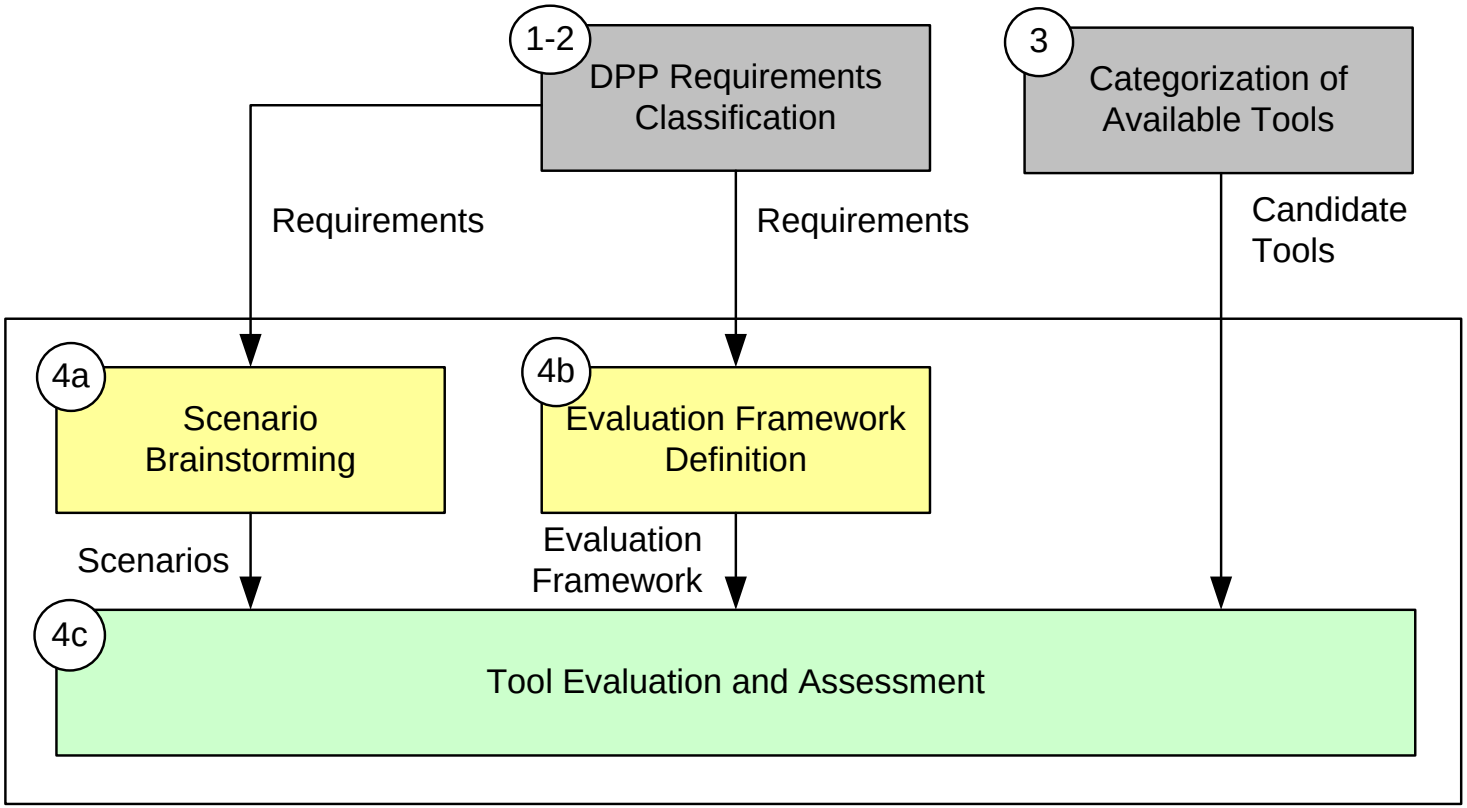
- Sample snapshot of candidate tools depending on the availability of tools.



Step 4: Tool Assessment: Evaluation Framework

→ Evaluation framework for systematically assessing candidate tools with respect to classified requirements.

- 4a. Identification of success-critical evaluation scenarios.
- 4b. Evaluation framework definition.
- 4c. Evaluation and assessment of tools based on captured scenarios.



Step 4a: Tool Assessment: Identification of Success-Critical Scenarios

Scenarios:

- Typical workflows and tasks based on user requirements.
- Guidelines for real-world tasks
- Scenario brainstorming workshop for DPP application

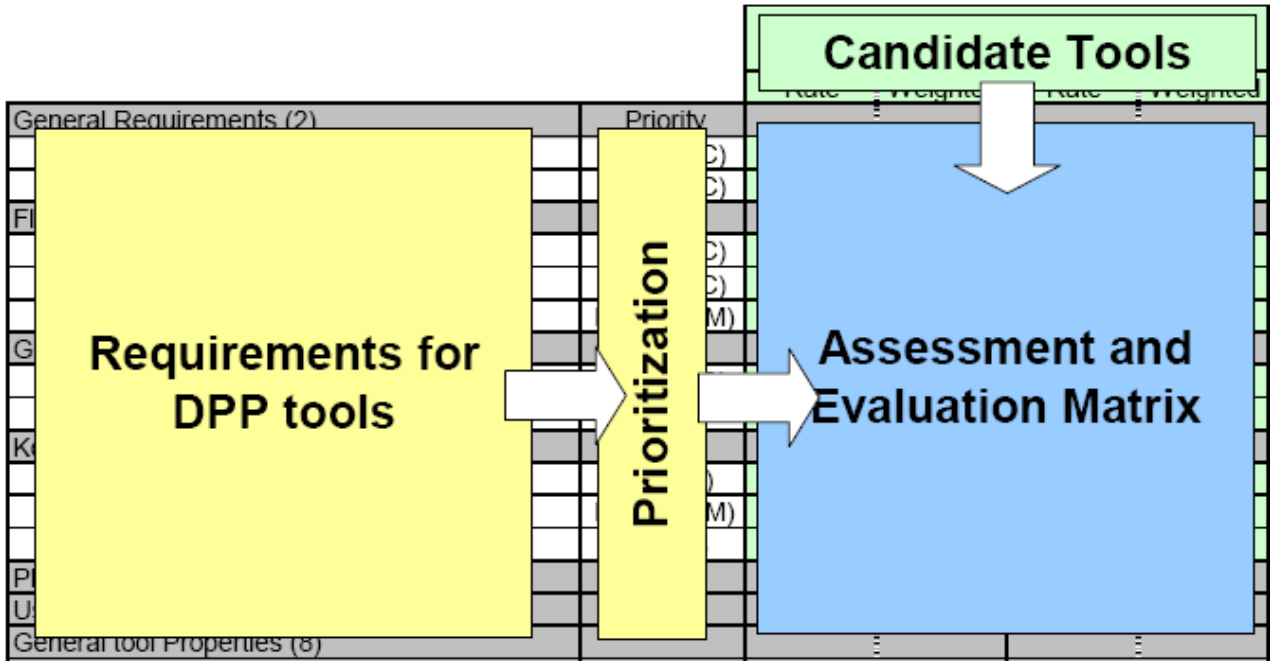
Selection of 6 basic scenarios:

- General scenarios (applicable to various types of tools):
  - Tool installation and configuration.
  - Tool performance.
- DPP specific scenarios
  - Initialization of a DPP session
  - Support of role assignment changes (Floor control).
  - Session Management (Storing/Restoring sessions).
  - Sample application for executing a small implementation task.

- Tool evaluation is based on scenario application and a subjective assessment of the Pair Programming Pair (Team result after discussion).

Step 4b: Tool Assessment: Evaluation Framework Definition

- Classified requirements and priorities (y-axis)
- Candidate tools for planned evaluation (x-axis)
- Tool evaluation matrix:
  - Estimation of the degree of requirements coverage by the tool:
    - Likert-scale: 0.. not supported / 5 ... fully supported.
    - 0/1 estimation if applicable, e.g., support of an individual platform (yes/no).
  - Weighting of the subjective assessment (acc. to requirement prioritization)



Step 4c: Tool Assessment: Snapshot of the Evaluation Matrix

- Tool application based on defined scenarios executed by real distributed pairs.
- Subjective Team assessment during/after tool application based on the classified requirements according to the evaluation framework.

			Screen Sharing				Collaborative Work			
			VNC		Netmeeting		Copper		PEP	
			Rate	Weighted	Rate	Weighted	Rate	Weighted	Rate	Weighted
General tool Properties (8)										
General Requirements (2)		**Priority**								
	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	5	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	2	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
Plattform (3)										
Usability (10)										

Step 4c: Tool Assessment: Aggregating Evaluation Results

- Aggregation of individual tool results to requirements categories (summarizing individual ratings).
- Threshold of a 3-level assessment based on the maximum value per requirements category.
  - Little tool support of attribute/requirement: 0-33% (red marked).
  - Medium support (33-66% (orange marked).
  - Comprehensive support: 66-100% (green marked).

	Maximum		Screen Sharing				Collaborative Work			
			VNC		Netmeeting		Copper		PEP	
	Rate	Weighted	Rate	Weighted	Rate	Weighted	Rate	Weighted	Rate	Weighted
General Tool Properties (8)	16	16	7	7	11	11	5	5	8	8
General Requirements (2)	10	100	9	90	8	80	8	80	5	50
Floor Control (7)	35	155	7	17,5	11	65	25	137,5	20	102,5
Gesturing (4)	20	92,5	9	38	9	27,5	2	10	1	5
Kommunikation (5)	25	60	0	0	18	50	6	17,5	7	17,5
Plattform (3)	3	6	2	5	1	2,5	3	6	3	6
Usability (10)	50	222,5	35	170	29	154	36	185,5	35	161,5
Total (39)	159	652	69	327,5	87	390	85	441,5	79	350,5

Results:
Evaluation of Selected Candidate Tools

- Share of requirements coverage (per category) based on the weighted assessment results.

	Category		Requirement Classification							Total		
Tool	Screen-Sharring	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	282	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 reacheable Assessment values:	159	652	

Conclusion & Further Work

- Applying DPP requires a strong tool support with focus on specific requirements for communication, interaction, data exchange, and collaboration.
- Evaluation frameworks, considering requirements classes, tool categories, and scenarios can help to assess candidate tools systematically.
- The results showed strengths and weaknesses and can be the starting point for further development of tools to efficiently support DPP in a distributed environment.
  - No tool under investigation supported DPP without limitation.
  - Strong benefits for the top-2 tools, especially designed for DPP.
  - Screen-sharing application can also support DPP to some extend.
- The proposed evaluation framework can support
  - Project managers and developers in selecting appropriate tool for project application.
  - Tool vendors to identify improvement options for DPP.
- Future work includes
  - Improvement and evaluation of the proposed process, i.e., refinement of requirements.
  - Extending the number of tools (including commercial tools).
  - Pilot application of most promising tools in real world project to get feedback from industry on the evaluation framework.

Thank you ...

Evaluating Tools that Support Pair Programming in a Distributed Engineering Environment

Dietmar Winkler, Stefan Biffl, Andreas Kaltenbach

Vienna University of Technology
Institute of Software Technology and Interactive Systems

http://qse.ifs.tuwien.ac.at
Dietmar.Winkler@tuwien.ac.at