Scrum Wall vs. Issue Tracker: 1-1

Mascha Kurpicz and Erwann Wernli

Software Composition Group
http://scg.unibe.ch

Abstract. Scrum is a lightweight iterative process that favors interaction between team members. Software development is however a complex activity and there exist many software tools aimed at supporting it. This research studies the role of software tools within Scrum practices. It focuses more specifically on comparing the strengths and weaknesses of the Scrum Wall and issue trackers, as they are frequently used together within projects. This paper presents findings from interviews that have been further validated with a survey. Results show that the Scrum Wall is highly appreciated by Scrum practitioners. It encourages positive dynamics and supports well most of the work organization. People tend to consider software tools as impediments, but use them nevertheless to control information that would otherwise remain tacit. Synchronizing information across tools is reported to be a source of troubles.

1 Introduction

Scrum is an iterative process that is now widely adopted in practice. Recent studies show that Scrum is even the most popular approach for agile development [15]. One of the pillars of Scrum is transparency [12] witnessed by its promotion of direct and informal communication between team members. For instance, the main event is the Daily Scrum during which the Development Team together plans the next 24 hours. The plan is reflected on the Scrum Wall, which is visible to all and which can be updated anytime.

Software development is a complex activity. Many software systems exist and are commonly used to support development teams in their work, such as wikis, issue trackers, portals, and so on. This paper aims at better understanding the relationship between traditional development tools and Scrum practices.

We organized our research into two parts. The first part focused on the following research questions (RQ):

1. What software tools are used by Scrum teams?
2. Why are these tools needed in the context of Scrum?
3. What are deficiencies of existing tools?

In order to answer these first questions, we interviewed 10 practitioners about their experience with Scrum projects. These interviews revealed that the most relevant tool is not a piece of software, but the the Scrum Wall. While the Scrum Wall was reported as a very effective way to collaborate, other software tools were still used by most teams for various reasons, such as legal requirements or to support distributed teams. We identified three main tool categories: (i) physical boards (analog), (ii) issue trackers (software), and (iii) document management systems (software). Linking and synchronizing information across tools was reported as a problem.

We decided in a second phase to better understand the strengths and weaknesses of the various tool categories (including the Scrum Wall). We focused on the following research questions (RQ):

4. How important are the various high-level needs of the team members?
5. What features are frequently used for each tool category?
6. What high-level needs does each tool category satisfy?

Our aim was to define stable dimensions and to build *profiles* for the different tool categories so that tools could be compared with each others. To do so, we analyzed the information collected during the interviews and identified on one hand a list of 8 recurring needs, and on the other hand a list of 9 recurring features. The needs and features were high-level and applicable to each tool category. We ran a multiple choice survey with 19 participants of different backgrounds. They were first asked to assess the relevance of the needs. Second, they had to assess their satisfaction with each tool category taking the list of features into consideration. Running surveys for all three tool categories would have resulted in a survey that would have been too long for our target audience[1]. In this paper, we provide detailed evaluations for only two tool categories: physical boards and issue trackers.

The key findings of this paper are the following:

- The Scrum Wall is highly valued by Scrum practitioners. They tend to perceive software tools as impediments: "the fewer tools, the better".
- In addition to the Scrum Wall, other software tools are used in most projects. The justification for these tools varies, *e.g.,* legal requirements, or support for distributed teams.
- Linking and synchronizing information across tools poses problems.
- The Scrum Wall facilitates reaching of consensus and taking collaborative decisions, which are important needs.
- Maintaining good documentation and conserving long-term knowledge are important needs, but neither the Scrum Wall nor software tools are adequate for this purpose.

These findings identify possible areas of improvements in software tools. These results are valuable not only for tool builders, but also for Scrum practitioners, to make more educated choices of tools.

The paper is structured as follows: section 2 presents existing tools and means to collaborate; section 3 describes the interviews; section 4 describes the survey and presents the profiles of the tools; section 5 discusses how software tools might look like in the future, and also their inherent limits; section 6 mentions threats

[1] The survey was limited to a maximum of 10 minutes by our industrial partner.

that could impact the validity of our research; section 7 concludes and presents our future plans.

2 Existing Tools

Scrum Support. In May 2010 the Forrester Wave published a paper that evaluated agile development management tools from various vendors [15]. Based on existing research, assessments of user needs, and interviews with vendors and experts they developed a comprehensive set of 152 evaluation criteria and evaluated the tool sets of various vendors against them. 117 of these evaluation criteria were summarized as "current offering" and included project setup, project and portfolio planning, project execution, project reporting and process customization. They found that IBM and MKS led the pack with the best overall current feature set [15]. IBM focuses on collaborative development and adds strong project management and analytics. According to their studies, MKS provides extensive task and workflow management and a good life-cycle integration.

L.S. Moller *et al.* propose a "Scrum tool for improving Project Management" [10]. Based on personal interviews they derived the following design requirements: an intuitive user interface, high accessibility, commitment to Scrum and having a project history. Following these requirements they developed an application in order to support the Scrum team and improve the project's management.

There are many other software tools that intend to support Scrum specifically. The two main artifacts of Scrum are the Product Backlog and the Sprint Backlog. Most of the tools focus on the Product Backlog rather than the Sprint Backlog. A comprehensive list of such tools can be found under `http://www.userstories.com/products`. The list includes not only dedicated tools, but also plug-in, templates, or extensions to popular development tools.

Scrum Wall. The Scrum Wall is normally realized as a physical wall covered with post-its and cards that can be moved. The Scrum Wall documents the state of the Sprint Backlog and belongs to the Development Team, which updates it each morning during the Daily Scrum or anytime a change occurs. The Scrum Wall makes the current progress of the sprint visible to the developers and the Scrum Master, and also to the Product Owner and possibly other stakeholders. Sutherland *et al.* [13] reported that a Scrum Wall in the office provides transparency and visibility. Several online services intend to provide "virtual boards" where cards can be moved, *e.g.,* Wallsome[2] and Trello[3].

Whiteboard. Whiteboards are frequently used by developers to support activities other than work organization. For instance, Cherubini *et al.* [2] investigated how and why software developers create diagrams. They observed that the majority of diagrams were sketched on whiteboards during ad hoc meetings. They consider the whiteboard to be attractive because it is ubiquitous and easy to use.

[2] `http://www.wallsome.com/`
[3] `https://trello.com/`

Touch Devices. Multi-touch devices enable new forms of collaborations. Among the many research projects in this category, Microsoft Surface grew into a commercial product that sees and responds to touch and real world objects, supporting more than 50 simultaneous inputs[4]. Based on Microsoft Surface, the Scrum Table[5] is a touchscreen that integrates with the Team Foundation Server. Such a tool has the advantages of a software tool (*e.g.,* updating and tracking of information) and a Scrum Wall (*e.g.,* touch-feeling or moving the cards on the wall). Harris *et al.* [8] compared multiple-touch and single-touch surfaces and investigated their potential to support children's collaborative learning interactions. Their results show that when using a single-touch device, children talked more about turn taking instead of the task at hand, while when using a multiple-touch device, they talked more about the content of the task.

Collaboration Support. Gutwin and Greenberg [7] present and define the concept of workspace awareness as "...up-to-the-minute knowledge about others' interaction with the workspace". They explain that it is "...part of the glue that allows groups to collaborate efficiently". They furthermore state that in a non face-to-face environment the designer of a groupware must recreate conditions and cues to allow people to derive workspace awareness. They designed a framework to address the questions of what information a groupware system should capture and how this information should be presented. Dourish and Bellotti [5] provide a similar definition of such an awareness defining it as "...an understanding of the activities of others, which provides a context for your own activity".

Collaborative Development Tools. Integrated Development Environments (IDE) are evolving towards Collaborative Development Environments (CDE). Booch and Brown [1] define a CDE as "...a virtual space wherein all the stakeholders of a project — even if distributed by time or distance — may negotiate, brainstorm, discuss, share knowledge, and generally labor together to carry out some task, most often to create an executable deliverable and its supporting artifacts". Omoronyia *et al.* [11] studied awareness in distributed software projects. They distinguish different kinds of awareness, defining context awareness as "...the evolving internal and external state information that fully characterizes the situation of each entity in a shared environment." They consider different ways of deriving awareness and present different tools for each category. For example, they propose social tagging — "the collaborative activity of marking shared content with tags as a way to organize content for future navigation, filtering or search" — as a way to derive awareness. This is supported by Jazz, a real-time team collaboration platform built on top of Eclipse IDE (for more information see [6]). Omoronyia *et al.* [11] state that "...addressing context awareness seems critical to the development of successful distributed collaborative systems, and the failure to do so effectively may be a key reason for lack of success to date".

[4] `http://www.microsoft.com/surface/en/us/default.aspx`

[5] `http://ifs.hsr.ch/ScrumTable.7133.0.html`

3 Interview — Understanding Problems with the Tools

3.1 Method

In a first step we wanted to address our three first research questions: What software tools are used by Scrum teams? Why are these tools needed in the context of Scrum? What are deficiencies of existing tools? To do so, we conducted interviews with Scrum practitioners to assess their experience with Scrum, the Scrum Wall, and software engineering tools. Ten experienced persons were interviewed for about 1 hour to 1 hour and 30 minutes. The interviews were structured the same way each time, and followed an interview script. The script had been designed to encourage people to talk, recall details of the project, and share their experience with us. If necessary, the scenario was of course adapted during the interview to match the experience and background of the interviewee. It was slightly refined after the first interviews, but its main structure was however stable.

First came an introduction during which the interviewee explained the scope of their project as well as the Scrum process they were following. Only after the introduction did we orient the interview towards questions explicitly related to tools. The aim was to understand whether developers used the Scrum Wall, whether and why they were using any additional tools, and what were the main problems raised by tools.

During the introduction, we asked the interviewees to describe how their Scrum process differs from the theoretical Scrum process; to describe the roles of the various people involved in the project; to describe the way they handle non-functional requirements; to recall what they changed in the way they worked since the beginning of the project; and to describe how their team/project fits in the broader context of corporate organization and product development.

During the phase concerning tools, we asked the interviewees to list all tools they are currently using; to list for each tool the positive and negative points; to list all media they used to track information about the project (from post-its to corporate portals); to describe their change management procedure; to describe whether they faced issues in updating written information and disseminating news; to describe how they tracked and triaged bugs; and to describe what they wish was automated. In case they were using the Scrum Wall and additional software tools for the same information, we asked them about the reasons for this redundancy.

3.2 Results

What tools are used (RQ1). Table 1 lists the tools mentioned during the interviews. We organized the tools into three categories:

Integrated tools that support tracking work items and managing documentation (*e.g.,* Microsoft Team Foundation Sever) are classified under both categories at the same time. We excluded tools for source code management, tools for continuous integration, and development environments (*e.g.,* Eclipse). We did not

Category	**Tool(# mentions)**
Physical boards *Boards on which we can draw and stick post-it*	Scrum Wall (9) Whiteboard(1)
Issue trackers *Systems to track work items in the broad sense of it,* e.g., *issues, bugs, stories*	MS Excel(9) Team Foundation Server(4) Trac(3) HP Quality Center(2) MS Project (1) OpenERP(1) Mantis(1) Sourceforge Enterprise Edition(1)
Document management systems *Systems to track file-based documentation*	Team Foundation Server(4) Subversion(3) Enterprise Architect(2) HP Quality Center(2) Clearcase SCM(1) MS Sharepoint(1) Sourceforge Enterprise Edition(1) OpenERP(1)

Table 1. The three tool categories we identified, and the number of mentions of each tool during the interviews.

have enough accurate information about them, since, several interviewees (in particular from Management) did not remember the exact name of the tools that developers were using. Subversion is however in the list as it was used not only for code, but also for document management. Tools developed in-house were mentioned three time during the interviews: a web-based database interface designed for the Scrum process, a tool to link tests and requirements together, and an extension of Microsoft Access. Unfortunately, interviewees did not remember sufficient details about them and we therefore excluded them from the list.

Why tools are used (RQ2). Most interviewees felt that "the fewer [software] tools, the better". They considered human interactions in front of the Scrum Wall to be an important aspect of Scrum, and software tools were rather perceived as impediments. Using only the Scrum Wall is however not realistic, and in almost all projects software tools were used. Various reasons to justify the usage of a tool in addition to the Scrum Wall were given: 5 out of 10 interviewees used a tool for reporting, four of them used either Excel and/or TFS and one team used a in-house developed tool; three interviewees belonged to distributed teams and used a tool to share information, in two cases they used an issue tracker and in one case they used a in-house developed software; in two interviews legal requirements (medical software in one case and banking software in the other case) constrained the team to use an issue tracker in order to provide additional project documentation.

Deficiencies of existing tools (RQ3). Two problems were reported as particularly important during the interviews. First was the problem of tacit knowledge sharing: five interviewees mentioned that most of the knowledge was in people's heads and they lacked project documentation. Substantial knowledge is lost when somebody leaves the team. Interestingly, according to one of the interviewee, "with the waterfall process the knowledge was in the head of one person, with Scrum it is in the head of the team". However, the problem remains basically the same. Tools fail to capture the tribal knowledge of development teams.

Second was the problem of overlap between tools: 6 out of 10 interviewees mentioned there was an overhead to synchronize information from the Scrum Wall to other tools (usually an issue tracker). After stand-up meetings around the Scrum Wall, the changes had to be manually synchronized in the software tools to reflect the Scrum Wall. Also, 3 of 10 interviewees mentioned that linking of information across tools was as an issue. They often had problems finding relevant information for their daily work across the different tools. Tools fail to integrate seamlessly with each other.

4 Survey — Building Tool Profiles

4.1 Method

In the second phase of our work we wanted to obtain more details about the strengths and weaknesses of each tool category, in accordance to the following research questions: What are the high-level needs that team members have? What are the typical features that tools offer? How frequently are these features used for each tool category? What high-level needs does each tool category satisfy? Our aim was to build *profiles* for the tool categories identified previously. Profiles would use the same dimensions in a way to be comparable against each others.

We identified recurring topics in the interview transcripts. We identified the following 8 recurring user needs:

1. Reach consensus / take collaborative decisions
2. Exchange information
3. Retain long-term knowledge
4. Assess progress
5. Organize and track time of work/duty
6. Know what I should do
7. Get accurate and trustable documentation
8. Know what the others are doing

We also identified the following 9 recurring features:

1. Enter new work item
2. Organize existing work items
3. Generate dashboard/Get an overview of the project status
4. Find relevant information for daily work in the issue tracker/Scrum Wall

5. Locate relevant information for daily work in other systems
6. Browse/see changes/history of a work item
7. Update the work items to latest decisions
8. Be notified about changes
9. Annotate, refine or comment a work item

The list of needs and features were high-level and could be applied to all tool categories, possibly with minor adaptations. For instance, "work item" can be a bug, a document, a user story or an issue depending on the tool of interest[6].

We designed a survey that assessed the relevance of the needs and assessed the satisfaction with each tool category. A requirement from our industrial partner was that the survey needn't more than 10 minutes to be filled. To keep the survey short, we limited it to the Scrum Wall and issue trackers, leaving the investigation of Document Management System for future work. The online survey was sent to 100 persons all involved in Scrum projects (developers, Scrum masters, managers). We obtained 19 results.

The exact structure of the survey was the following: in a first part, participants were asked to order the 8 needs according to their importance. This information reveals the priorities of the teams for successful collaboration. In a second part, we asked them how frequently they used the features of their issue tracker and the Scrum Wall. This reveals the main features of each tool category. Needs that must be fulfilled for the user's satisfaction often cannot be implemented directly as features, for example, that the user needs "to be aware of what's happening". Therefore we must link the user's needs to the tool's features to emphasize the relations and dependencies between them. In a third part, we asked them to evaluate their satisfaction with the tools with regard to the needs mentioned before. This reveals important needs with a low satisfaction, where further effort is required.

Finally, we asked them whether they considered synchronizing and linking information across tools to be an issue. This point had indeed been raised frequently in the interviews and we wanted a confirmation of its importance. At the end was also a free text area where participants could provide their feedback on the survey.

4.2 Results

The survey used likert-type scales. We present the results with box plots showing the lower quartile, median, upper quartile and the mean, indicated with a cross.

Importance of the high-level needs (RQ4). Figure 1 shows the results for the first part of the survey focused around the importance of needs.

The important needs represent direct collaboration: exchanging information amongst collaborators, taking decisions together, and being aware of ongoing work to perform. Needs considered less important imply taking some distance

[6] The exact wording of the survey can be found at `http://scg.unibe.ch/wiki/students/mkurpicz/ToolsupportforScrum`

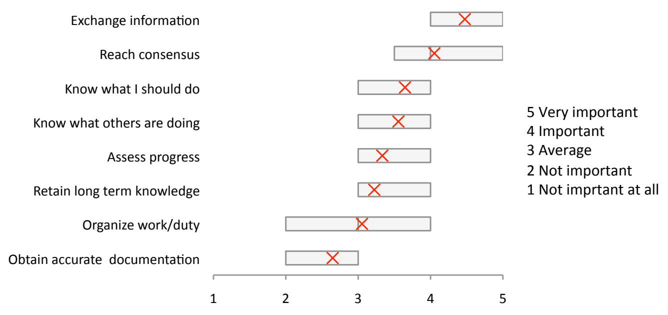

Fig. 1. Importance of the needs.

with the immediate work on the project: assess the progress, write down tacit knowledge, organize work. Needs related to daily work are more important than needs related to the long-term perspective of the project.

The exchange of information was considered the most important need: 12 of 19 persons considered it very important, 4 considered it important and the rest as average. Obtaining accurate and trustable documentation is however not perceived as very important. Probably there isn't too much emphasis on written documentation. These results indicate that sharing knowledge in the team is important, but happens informally.

Reaching consensus is the 2nd most important need. "One of the pillars of Scrum is that once the Team makes its commitment, any additions or changes must be deferred until the next Sprint." [4] The team decides as one entity and shares the required knowledge for that decision within the team. Scrum shifts the focus from individuals to the whole team and these results highlight the importance of collaborative decision taking.

Frequency of use of the features (RQ5). Figure 2 shows the results of the survey concerning how frequently the features were used. The top 3 features are highlighted in grey. The charts mention the exact wording used in the survey.

The Scrum Wall is frequently used as a tool that gives an overview of the progress (the feature "Generate Dashboard/Report" was phrased "get an overview of the project status" for the Scrum Wall in the survey). It seems also to be an effective way to capture the information people really need for the day-to-day work. Information is easier to extract from the Scrum Wall than from the Issue Tracker.

Features related to managing changes were used more frequently in the case of the Scrum Wall. People annotate, refine, comment, update it more frequently than an issue tracker. They are also informed about changes more frequently with the Scrum Wall than an issue tracker (9 participants used the Scrum Wall to be notified about changes 1-5 times per week or more, while 6 of them stated the same for the issue tracker). Possible explanations for this are the very intuitive nature of the Scrum Wall, and the fact that people easily notice when somebody

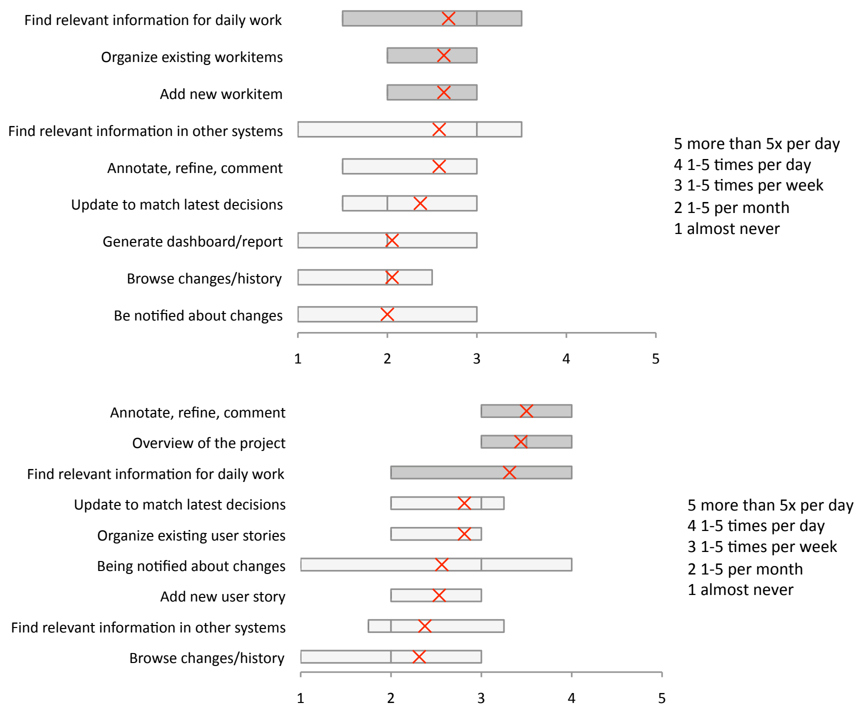

Fig. 2. Frequency of use of the features for the issue tracker (top) and Scrum Wall (bottom).

alters the Scrum Wall due to its physical proximity. The Scrum Wall enables dynamism, and favors quick updates.

Satisfaction of the high-level needs (RQ6). Figure 3 shows the satisfaction of users according to the previously identified needs. The Scrum Wall shines in fulfilling most needs, but fails completely at others. The issue tracker fulfills most of the needs, but but to a lesser degree.

The Scrum Wall supports and favors the exchange of information extremely well. It is also excellent for increasing awareness, *i.e.,* "know what others are doing". The Scrum Wall acts as an "information radiator" within teams [3].

It fails however to keep any kind of long term information. Clearly, this is not its intent, and this highlights the very dynamic and transient nature of information on the Scrum Wall. The need for accurate documentation is however not really fulfilled any better by the Issue Tracker. This corroborates with feedback in the interviews: it was often mentioned that it is "a problem to have documents up to date for everybody". It would be interesting to profile document management systems to assess whether they effectively fill this gap. We plan to do this in future work.

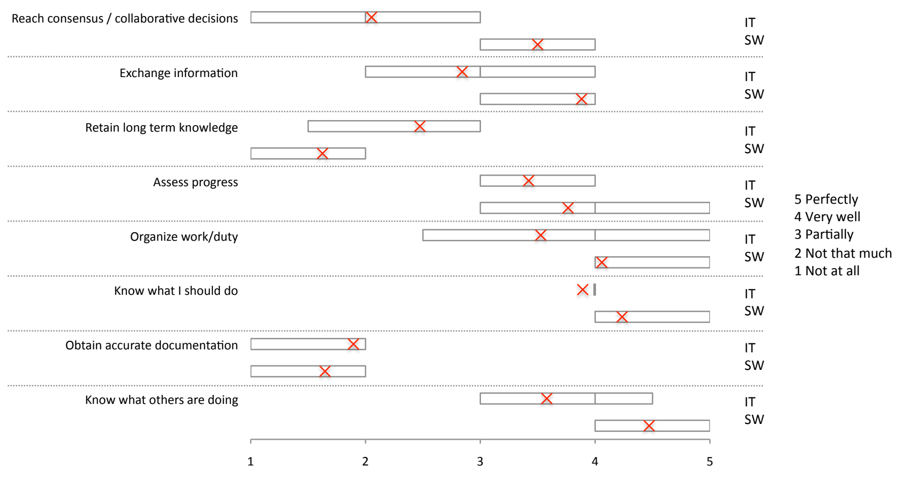

Fig. 3. User satisfaction for the issue tracker (IT) and Scrum Wall (SW). In each group, the issue tracker (IT) is at the top, and the Scrum Wall (SW) at the bottom.

Areas of improvement. Correlating the importance of the needs with how well they are satisfied, we obtain two matrices, respectively for the Scrum Wall and Issue Tracker (See Table 2 and Table 3).

Table 2. Areas of improvement for the Scrum Wall

Scrum Wall	**Important needs**	**Less important needs**
High Satisfaction	Exchange of information Know what I should do Know what the others are doing Reach consensus Take collaborative decisions	Assess progress Organize work/duty
Low Satisfaction		Accurate and trustable documentation Retain long-term knowledge

Table 3. Areas of improvement for the issue tracker

Issue tracker	**Important needs**	**Less important needs**
High Satisfaction	Know what I should do Know what others are doing	Assess progress Organize work/duty
Low Satisfaction	Reach consensus Take collaborative decisions Exchange of information	Accurate and trustable documentation Retain long-term knowledge

Table 3 indicates that software tools should be improved to better support collaborative decision taking in order to compete with their analog counterparts. Improvements in this area would be highly valuable for distributed teams that do not have the possibility to meet physically, yet still need to collaborate.

The conservation of long-term knowledge and keeping documentation up-to-date are long-standing problems in the field of software engineering. Both tables confirm this once more [2].

The survey also confirmed the impression we had from the interview that synchronizing and linking information across tools is a problem worthy of further investigation: 9 out of 19 participants considered linking information across tools to be problematic; 15 out of 19 participants considered the synchronization of information to be an issue.

5 Discussion

Scrum teams organize their work around the Scrum Wall by writing, moving and annotating cards. This was perceived to be a key ingredient to the success of Scrum. Scrum does *not* need complex software tools and procedure, only pen and paper. In the free text area in the survey, 6 of the 19 participants mentioned explicitly that they prefer Scrum without any software tools. This striking result shows how human factors are central in Scrum. The overhead and lack of transparency of software tools is seen as an impediment.

"We don't need tools for Scrum" is probably true for the development team. However, the team does not work in isolation, and it needs to interact with external actors. In most projects (8 of 10 interviews) the team used an additional software tool for one reason or the other. For example, due to legal requirements in the development of a medical software system, a team had to deliver formal documentation and used Sourceforge Enterprise Edition to manage written documentation. In another project, the Scrum team was part of a broader non-agile project. The non-agile surrounding context forced them to adapt the process and use an extra software tool to deliver reports periodically. The need for reporting and documentation arises because of interactions with other teams outside the Scrum team.

Unfortunately, using multiple software tools introduces other problems, one of which is the overhead to synchronize information across tools. One way to overcome this problem is to use an integrated platform like Microsoft Team Foundation Server. This however comes with an increased complexity. The setting of each development team is different — actually a lot more different than we expected — and tailoring such an integrated environment requires specialists. The growing interest in the social dimension of development is visible in the emergence of collaborative development environments. These tools extend traditional development environments with additional channels that increase awareness. Another area of active development are touch devices such as the Microsoft Surface. These products enable collaboration and bridge the gap between the analog and digital world. These different areas of innovation can be combined. The Scrum Table is for instance a tactile front-end to TFS based on Microsoft Surface. This leads us to wonder what will be the development tools of tomorrow, and whether they will reconcile high dynamism with formal reporting and documentation.

The quote “a fool with a tool is still a fool” was cited multiple times by participants in the interviews and the survey. Clearly, some problems cannot be resolved by tools and people are aware of this fact. For example, if the root problems of a team are interpersonal issues and bad internal communication, no tool will ever help. Tools can be an important factor for the success of a project if they improve the information flow, but can also turn into liabilities.

6 Threats to Validity

We conducted qualitative interviews and ran a follow-up survey to confirm and detail our findings. The data collected were grounded in practitioner’s experience. However, the collection and analysis of data might have been subject to several threats that might impact our conclusions.

Internal validity Obviously, the role of the interviewer is significant during qualitative interviews. The interviewer conducts the discussion towards certain themes and she might be influenced by her personal experience. It must be noted regarding this point that the interviewer was a computer science student with little professional experience. This is an asset compared to professional developers or experienced researchers for whom it might be harder to remain unbiased. The interpretation of the data is influenced by the researchers’ experience as well. The transcripts of the interviews were coded [14] once by the interviewer, then discussed with the co-author. The themes that emerged were consistent.

It is well known that answers to surveys are sensitive to personal interpretations and that exact phrasing is important. The survey was designed to collect facts and mitigate such effects. The exact phrasing of the features (see RQ5) had been slightly adapted to the specificities of each tool[7], yet remained comparable. We took care however to design the survey so that it did not favor one tool over the other. The role of the participants (developer, Scrum master, manager) might explain certain variations in the answers. The variations were however not significant enough to account for the role in our analysis. Lastly, it is worth noting that we collected the participant *belief* on how frequently they used certain features. To collect objective frequencies, one would need to *observe* the actual behavior of practitioners in their workplace. We asked for the frequency of usage rather than the importance (as we did for other questions) because we assumed it is more objective.

External validity The survey participation was 19%. The answers we obtained might not be representative for the whole population of Scrum practitioners.

[7] see `http://scg.unibe.ch/wiki/students/mkurpicz/ToolsupportforScrum`

The participants and interviewees were selected from different groups in a software company of more than 400 employees. The corporate culture might have impacted our results, though.

7 Conclusion

Scrum favors direct and informal communication, notably during sessions around the Scrum Wall. The Scrum Wall supports well the informal day-to-day activities of the development team. It is highly valued by practitioners and perceived to be an effective tool to exchange information and raise awareness. Such informal work organization has limits though. Formal control might be required (*e.g.,* reports) and formal documentation might need to be produced (*e.g.,* legal constraints). Issue trackers and document management systems are frequently used in addition to the Scrum Wall, mostly for these two reasons. These findings come from qualitative interviews with Scrum practitioners.

Different tools have different characteristics. We focused on a second part our research on the strength and weakness of the Scrum Wall and Issue Tracker. We conducted a survey and built a profile for each category of tool. The Scrum Wall excels at providing an overview of the project status, is an effective tool to find information relevant for the daily work, and is very easy to update. It fails however at keeping track of the long-term perspective of the project. The profile of the issue tracker is more flat. It supports relatively well organizing the day-to-day work, and supports some form of knowledge retention. Needs related to the daily work were assessed as more important than needs related to the long-term perspective of the project.

Taking collaborative decisions is one of the highest needs that team members have. This research confirms that physical proximity, as with the Scrum Wall, favors such collaboration. It also confirms that knowledge is frequently tacit and that documenting software is a significant problem. Lastly, it indicates that using several tools together leads to overlap between tools, and problems to synchronize and link information across tools.

We believe that software tools of tomorrow will offer the best of both worlds: the human factors of the Scrum Wall and the possibility to conserve knowledge at the same time. As a first step to improve the status quo we have implemented a prototype of a tool that synchronizes the Issue Tracker based on a picture of the Scrum Wall [9]. We plan in the future to turn the prototype into a working product, and to build profiles for document management systems.

Acknowledgments We want to thank Zuehlke Engineering for their collaboration during the project. We would like to thank Mircea Lungu and Oscar Nierstrasz for kindly reviewing earlier drafts of our paper.We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project "Synchronizing Models and Code" (SNF Project No. 200020-131827, Oct. 2010 - Sept. 2012).

References

1. G. Booch and A. W. Brown. Collaborative development environments. 2003.
2. M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko. Let's go to the whiteboard: how and why software developers use drawings. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 557–566, New York, NY, USA, 2007. ACM.
3. A. Cockburn. *Crystal Clear*. Addison-Wesley Professional, 2004.
4. P. Deemer, G. Benefield, B. Vodde, and C. Larman. Scrum primer. *Scrum Training Institute*, 2010.
5. P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, CSCW '92, pages 107–114, New York, NY, USA, 1992. ACM.
6. R. Frost. Jazz and the Eclipse way of collaboration. *IEEE Softw.*, 24:114–117, November 2007.
7. C. Gutwin and S. Greenberg. Workspace awareness for groupware. In *Conference companion on Human factors in computing systems: common ground*, CHI '96, pages 208–209, New York, NY, USA, 1996. ACM.
8. A. Harris, J. Rick, V. Bonnett, N. Yuill, R. Fleck, P. Marshall, and Y. Rogers. Around the table: are multiple-touch surfaces better than single-touch for children's collaborative interactions? In *Proceedings of the 9th international conference on Computer supported collaborative learning - Volume 1*, CSCL'09, pages 335–344. International Society of the Learning Sciences, 2009.
9. M. Kurpicz. Tool support for Scrum. Bachelor's Thesis, Software Composition Group, University of Berne, 2011.
10. L. S. Møller, F. B. Nyboe, T. B. Jørgenson, and J. J. Broe. A Scrum tool for improving project management. *Flirting with the Future, Prototyped Visions by the Next Generation, Proceedings of the Fifth Student Interaction Design Research Conference (SIDeR '09)*, pages 30–32, 2009.
11. I. Omoronyia, J. Ferguson, M. Roper, and M. Wood. A review of awareness in distributed collaborative software engineering. *Software: Practice and Experience*, 40(12):1107–1133, 2010.
12. K. Schwaber and J. Sutherland. The Scrum guide, 2001.
13. A. C. Sutherland, J. Sutherland, and C. Hegarty. Scrum in church: Saving the world one team at a time. In *Proceedings of the 2009 Agile Conference*, AGILE '09, pages 329–332, Washington, DC, USA, 2009. IEEE Computer Society.
14. R. S. Weiss. *Learning From Strangers: The Art and Method of Qualitative Interview Studies.* Free Press, 1994.
15. D. West and J. S. Hammond. Agile development management tools. *The Forrester Wave Q2*, 2010.