A Case Study on Tool Support for Collaboration in Agile Development

Fabio Calefato
University of Bari
Bari, Italy
fabio.calefato@uniba.it

Andrea Giove
University of Bari
Bari, Italy
a.giove16@studenti.uniba.it

Filippo Lanubile
University of Bari
Bari, Italy
filippo.lanubile@uniba.it

Marco Losavio
Klopotek Software & Technology Services Italia
Gioia del Colle, Italy
m.losavio@klopotek.it

ABSTRACT

We report on a longitudinal case study conducted at the Italian site of a large software company to further our understanding of how development and communication tools can be improved to better support agile practices and collaboration. After observing inconsistencies in the way communication tools (i.e., email, Skype, and Slack) were used, we first reinforced the use of Slack as the central hub for internal communication, while setting clear rules regarding tools usage. As a second main change, we refactored the Jira Scrum board into two separate boards, a detailed one for developers and a high-level one for managers, while also introducing automation rules and the integration with Slack. The first change revealed that the teams of developers used and appreciated Slack differently with the QA team being the most favorable and that the use of channels is hindered by automatic notifications from development tools (e.g., Jenkins). The findings from the second change show that 85% of the interviewees reported perceived improvements in their workflow. Despite the limitations due to the single nature of the reported case, we highlight the importance for companies to reflect on how to properly set up their agile work environment to improve communication and facilitate collaboration.

CCS CONCEPTS

• **Human-centered computing** → *Empirical studies in collaborative and social computing.*

KEYWORDS

Slack, computer-mediated communication, Jira, collaboration tools, agile development, action research, case study

1 INTRODUCTION

Agile development is an increasing trend for software organizations, whether small or large, working co-located or geographically distributed [14]. Still, there are many challenges in supporting effective collaboration and communication for agile teams [10–12]. For example, an ever-growing plethora of different tools are needed to develop and manage software projects [2, 5], placing teams into a situation of information fragmentation [18] and overload [13].

This paper describes a longitudinal, single case study conducted at the Italian site of a large, globally-distributed company named Klopotek. Our research aims at studying the Agile work environment at Klopotek to identify points of friction between development practices and collaboration tools. As such, we are interested in answering the following research question: *How can the Agile work environment be improved to facilitate communication and better support the collaboration workflow?*

The case study lasted for about four months (from December 2018 to April 2019) and was executed into two consecutive steps. First, we gained an understanding (via direct observation and semi-structured interviews) of the work environment, that is, how development and collaborative tools are used. Then, in line with action research guidelines [1, 7], we introduced two main changes aimed respectively at improving the communication and collaboration workflow by optimizing how the Slack and Jira tools were used. The first change revealed that the teams of developers used and appreciated Slack differently, with the QA team being the most favorable and that the use of channels is hindered by automatic notifications from development tools (e.g., Jenkins). The findings from the second change show that 85% of the interviewees reported perceived improvements in their workflow.

The research presented here is first step focused on streamlining communication and collaboration at the Italian site of the company, before bringing and analyzing the changes to the other remote sites. As main contributions, in our paper we provide *(i)* recommendations on how to set up a Slack workspace for supporting Agile teams as well as an understanding of the (side) effects of *(ii)* integrating notifications from external development tools with channels and *(iii)* using multiple Jira boards with different perspectives and automation rules.

The remainder of the paper is organized as follows. In Sect. 2, we describe the case subject and the settings for the direct observation and changes. In Sect. 3 and 4, respectively, we report and discuss the results from our analyses, comparing our findings to related work. Finally, we conclude in Sect. 5.

2 CASE STUDY

In this section, we report about the mixed-method, longitudinal case study conducted on-site, following the guidelines provided by Runeson et al. [15, 16].

Accordingly, below we first introduce the company (Sect. 2.1.1). Then, we describe the data collection procedure used to gather a clear understanding of the work environment and tooling (Sect. 2.2). Finally, we detail the two changes introduced *in vivo* to improve communication and collaboration (Sect. 2.3).

2.1 Klopotek

Founded in 1992 in Berlin, Germany, Klopotek AG is now a global company operating in the area of publishing software, with sites in Europe and the USA. Klopotek works with more than 350 publishers worldwide, for a total of over 14,000 users in 140+ locations.

In 2015, the company opened near Bari, Italy, a site dedicated to development. *Klopotek Software & Technology Services Italia* (KSTS-It) was opened with the specific goal of redeveloping the legacy ERP desktop application into a cloud-based web application, called Stream. The other two projects on which KSTS-It works are Klopotek Deployment Manager (KDM), a tool used both internally and by customers to install and configure Klopotek solutions without manual intervention, and Core, a framework that allows both the web app and the legacy tool to interact with the back-end via a RESTful API.

2.1.1 Organization. KSTS-It is currently structured into five main areas, each corresponding to a specific team:

- **Architecture**. A team of senior architects in charge of the design and development of the new Core framework, while also taking care of the coexistence with the legacy solution.
- **Dev**. The team consists of seven developers taking care of all the development and maintenance activity on some Stream products catalog. Some of them also contribute to Core.
- **QA**. The team consists of three members, also associated with the Dev team, who take care of functional and non-functional testing, documentation, as well as process and product monitoring.
- **DevOps**. A team of software engineers who work closely with Dev, QA, and Architecture, respectively, to speed up the integration/deployment of new features/releases and the configuration of the testing and development environments.
- **Management**. This board includes the KSTS-It director, the leaders of each of the previous four teams as well as the Product Manager and the Technical Product Manager.

The KSTS-It site is located in a three-floor building. Due to the co-location in one large room, within-team interaction typically happens face-to-face (F2F), and developers sometimes decide to pair-program, if needed, as well as testers who often work in pairs with developers.

2.1.2 Development Process. The Dev and QA teams follow a hybrid process that is transitioning towards the Scrum methodology [17]. Instead, at the time of the case study, the Architecture and DevOps team had not adopted any specific methodology, but were considering the same idea of transitioning to Scrum as well.

Table 1: Tools used at KSTS-It, arranged by category.

Category		Tool
Development	SCM	GitLab
	Coding	Balsamiq, PapDesigner, Visual Paradigm, Eclipse, WebStorm, VS Code
	Issue Tracking	Jira
	Build	Maven, Grunt
	Testing	JUnit4, JMeter, HP Quality Center
	Inspection	SonarQube
	Documentation	Confluence
	Ticketing	BugTrace (in-house tool)
Infrastructure mgmt	CI/CD	Jenkins, KDM + Ansible
	Repo Management	JFrog Artifactory (internal), Sonatype Nexus (customers)
	Config/Provisioning	KDM
	Release Management	KDM + custom bash scripts
	Containerization	Docker
	Monitoring	KDM, Apache Kafka, ElasticSearch
	Logging	Logstash
Collaboration	Communication	Microsoft Exchange, Skype for Business, Slack

Overall, the development process involves three main types of activities: *analysis & design*, *development & test*, and *release & delivery*.

The *development & test* activities involve periodic Grooming and Sprint Planning sessions, organized typically on Monday afternoon or Tuesday morning. Here participants (i.e., the Dev team, the Dev team leader, and the Technical Product Manager) make estimations of user stories based on the development time (in person-days). Story points are assigned to user stories using the *Planning Poker* technique [4, 6].

Besides Grooming and Sprint Planning, also Daily and Sprint Review Meetings are conducted. The Daily Meetings are organized every day at 9:15, involving the Technical Product Manager and the team members of the Dev and QA teams (including the leaders of both teams). As for the Sprint Reviews, they also involve members from the Berlin site for better coordination and are followed by Retrospective Meetings, which instead involve only developers and testers, to give them the maximum freedom of expression. Here, the QA team is also responsible for assessing the quality of Sprints.

The activities of *release & delivery* change depending on the product but typically involve *Snapshot* releases updated daily, *Beta* versions released at the end of each Sprint, *Release Candidate*, i.e., the version under test before delivery, and *Final*, i.e., the delivered version.

2.1.3 Tooling. The teams at KSTS-It use a plethora of tools, as it can be observed in Table 1. The high number of adopted tools is a side effect of the coexistence of two solutions, the legacy desktop product and the new web application, which rely on different technology stacks.

The tools can be broadly grouped into three main categories, that is, *development*, *infrastructure management*, and *collaboration*. Next, we comment on the use of only the tools that are relevant to the presentation of the case study.

In the development category, Jira plays a pivotal role in handling the backlogs of the Dev, Architecture, QA teams, each with slightly different modalities. The Dev team relies on a Scrum board with user stories (broken down into sub-tasks) that transition into four

states (*To do*, *In progress*, *Fixed/Developed*, and *Done*). The DevOps team uses a Kanban board with six states (*To do*, *Planned*, *In progress*, *Done*, *QA*, and *Released*). Within the board, the team graphically groups tasks together to separate the high-priority one from those related to, for example, infrastructure maintenance. Finally, the Architecture team uses *Epics* to group related user stories but does not rely on any specific type of board.

Another important development tool is Confluence, a workspace that stores all the knowledge base of the Klopotek ecosystem, from QA reports and deliverables to FAQs and how-tos.

Among collaboration tools, KSTS-IT rely on Microsoft Exchange for email and calendar. Email is the primary means of communication, used mostly for asynchronous communication with the Berlin site, but also for more formal internal communication at KSTS-IT. Microsoft Exchange is also used because it integrates with Skype for Business, which is the preferred instrument for video- and conference-calls with Berlin, albeit it is sometimes used for instant messaging (IM) too.

Other than Skype, Klopotek has adopted Slack as the means of site-specific, technical-oriented communication. In fact, there are two separate workspaces, one for each of the two development sites. Regarding KSTS-IT, Slack is mostly used for quick questions, code snippet and file sharing, and one to one communication. Channels are instead used to gather information from external sources. At the time of the case study, the following integrations were already active: GitLab, Jenkins, Eclipse Code Sharing, and Google Drive.

2.2 Data Collection

Overall, the case study lasted for about four months, from December 2018 to April 2019. The collection of data happened along the entire duration of the case study. There were four main data sources, that is, *documentation*, *direct observations*, *tools*, and *interviews*, as detailed next.

2.2.1 Documentation. During the case study, we had access to documentation of different kind, such as PowerPoint presentations (describing the company's development process at a higher level, organization, and product architecture) and Excel spreadsheets (detailing project planning and performance analysis). Other helpful sources of documentation were the *Strengths, Weaknesses, Opportunities, Threats* (SWOT) [8] and *Balanced Scorecard* [9] analysis reports, which helped us understand the areas perceived by developers and management as susceptible to improvement.

Overall, the pieces of information retrieved from documentation sources were used to triangulate, integrate, and confirm those obtained through the direct observation and semi-structured interviews, as described next.

2.2.2 Direct observations. Regarding direct observations, we could attend 18 different sessions (see Table 2). The company allowed us to participate in one Grooming session, seven Daily Meetings, and two Sprint Reviews with the development teams. We agreed not to take part in Retrospective meetings to avoid the risk of limiting the free expression of problems reported by the development team members. We also had the chance to participate in six KSTS-IT Management Board meetings and two Technical Board meetings.

Table 2: List of the 18 formal sessions of direct observation.

Date	Session
05 Dec. 2018	Management Board Meeting
10 Dec. 2018	Management Board Meeting
11 Dec. 2018	Management Board Meeting
07 Jan. 2019	Grooming
08 Jan. 2019	Daily Meeting
09 Jan. 2019	Daily Meeting
09 Jan. 2019	Sprint Review
10 Jan. 2019	Daily Meeting
14 Jan. 2019	Daily Meeting
15 Jan. 2019	Daily Meeting
17 Jan. 2019	Daily Meeting
18 Jan. 2019	Daily Meeting
23 Jan. 2019	Sprint Review
15 Jan. 2019	Technical Board Meeting
24 Jan. 2019	Technical Board Meeting
31 Jan. 2019	Management Board Meeting
04 Mar. 2019	Management Board Meeting
25 Mar. 2019	Management Board Meeting

The first session that we could observe was the KSTS-IT Management Board Meeting held in Dec. 2018, when a first draft of the SWOT analysis report for the year 2018 was discussed. This opportunity turned out to be fundamental for the case study because it allowed us to get an immediate understanding of the internal problems and goals, and thus inform the rest of the observations.

Finally, we found it essential that Management allowed us to have one desk in the open-space office shared by all development teams, to directly observe how collaboration and communication happened daily.

2.2.3 Tools. Beside the desk in the open-space office, the managers allowed us to have credentials to access most of the tools used by the teams. We focused on Confluence, Jira, and Slack. Having access to the Confluence knowledge base was helpful in retrieving Sprint Review and Retrospective reports as well as obtain detailed information about the architecture and development of the products, testing, and CI/CD processes. Regarding Jira and Slack, the former was useful for understandings the workflow of the development teams (e.g., task assignments), whereas the latter was used to complement the direct observations and have access to the interactions that did not happen (e.g., chats).

2.2.4 Interviews. Along the case study, three sessions of semi-structured interviews[1] were conducted: *initial* (exploratory), *intermediate* (preparation), and *follow-up* (feedback).

Initial, exploratory interviews were conducted at the beginning of our case study, around mid-December, after participating in the first three KSTS-IT Management Board meetings (see Table 2). Since the intention was to clarify the results of the SWOT analysis, these interviews involved all the seven members of the Management Board. The interviews lasted about 15-20 minutes.

Intermediate interviews were conducted in late January. The interviews lasted about 15-20 minutes each and also involved members of development teams (Dev, QA, Core, and DevOps), for a total of twenty-two participants. In this case, the template for guiding the interviews was structured in three parts. The first part gauged the experience of the interviewees; the second one focused on how they used tools, in particular, Outlook, Slack, Jira, and Confluence;

---

[1]The scripts used in each interview session are available at https://figshare.com/articles/ICGSE_2020_Interviews_material/11481759
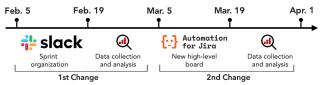
Figure 1: Timeline of the two introduced changes.

the third one contained different questions depending on the role of the interviewee, aimed at uncovering job-specific themes.

The follow-up interview sessions were conducted to collect post-change feedback. As detailed in the next Sect. 2.3, we introduced two changes, and feedback collection happened in mid-February and at the end of March. The goal here was to elicit feedback (e.g., effectiveness, appreciation) of the change introduced. After the first change, we interviewed Dev and QA team members (13 in total), whereas after the second, we also interviewed Core and DevOps team members (20 in total). Each interview lasted about 10 minutes and contained both closed- and open-ended questions.

2.3 Changes

In Fig. 1 it is illustrated the timeline of the two changes introduced at KSTS-Iᴛ. After initial observations and discussions with management, we learned that they (i) were seeking solutions to streamline communication, both inter- and intra-site, and (ii) wanted to make the best of Jira, which was chosen as the core of their collaboration infrastructure. For each change, we observed its effects along an entire Sprint iteration (i.e., two weeks), at the end of which feedback was collected using semi-structured interviews, to understand whether the change was to be accepted or rejected and if it needed adjustments.

The *first change* (Feb. 5 – Feb. 19) focused on communication. Its goal was to promote more rational use of the communication tools, in particular to foster the use of Slack for internal communication instead of email and Skype for Business, which instead had to be used for all the external communication (e.g., for messages and calls between KSTS-Iᴛ and Berlin or customers). In particular, three new public channels were added, #dev, #qa, and #sprints.

The expected benefit of the proposed change was making the internal conversations searchable and organized in proper channels, while also avoiding information to be fragmented over multiple channels and reducing the clutter of email inbox.

Regarding the execution, at the beginning of the selected Sprint iteration, a meeting was arranged with the members of the development teams involved in the change, i.e., Dev and QA. Here, the change was explained, and instructions were given, such as avoid using direct private messages in Slack for team-relevant communication. Eventually, we concluded the meeting with a Q&A session to resolve any doubts.

Other than the participant feedback collected through follow-up interviews, we retrieved communication logs from Slack to analyze both quantitatively and qualitatively the effects of the proposed change.

The *second change* (Mar. 5 – Mar 19) focused on the Jira tool and had a two-fold goal. First, empowering managers through the creation of a high-level board in Jira—to get a glimpse of the status of user stories—separated from the detailed board with sub-tasks assigned to developers and testers. Also, two new columns relevant

for the QA team were added to the new board for stories ready to be tested and those currently under test, thus generating the following state transitions: *To do*, *In progress*, *Read to test*, *Testing*, and *Done*. Second, through the integration of a Jira plugin,[2] we introduced automation rules to automatically map the states between low-level tasks in the detailed board and high-level user stories in the Management board. Also, the plugin allowed the notification of Jira events via Slack rather than email, for better integration between the two tools.

The expected benefits of the proposed changes were relieving users from the tedious duties of managing task states manually, increasing the visibility of important events via Slack, and reduce email overload.

Regarding the execution, at the beginning of the selected Sprint iteration, we arranged another preparation meeting, this time also involving the Management and Technical Board members, other than the other development teams. Here we illustrated the intended use of the new Management board as well as the automation rules offered by the Jira plugin and how to define custom ones.

Also, for the second change, we collected Slack communication logs along with follow-up interviews.

3 FINDINGS

In this section, we first report the findings from the direct observation and the initial/intermediate interviews (Sect. 3.1). Then, we report the findings from the two changes (Sect. 3.2).

3.1 Direct Observation and Initial/Intermediate Interviews

We conducted semi-structured interview sessions with twenty participants, in particular five developers (D1-D5), two testers (QA1-QA2), three members of the DevOps team (DO1-DO3), two members of the Architecture team (A1-A2), the four team leaders (one from each of the development teams, TL1-TL4), the and four managers (M1-M4).

Four participants are female (19%). Regarding the working experience, at the time of the case study, the managers and team leaders had been working at KSTS-Iᴛ for over three years, and had an overall working experience of 10+ years. The developers were less experienced: six had been working at KSTS-Iᴛ for less than a year and had an overall experience ranging between 1 and 5 years; the other six developers were more senior, with an overall working experience between 5 and 10 years, of which 1 to 3 years spent at KSTS-Iᴛ.

3.1.1 Communication Issues. One of our goals for the case study was to understand how communication tools were used. We directly observed that most of the internal communication at KSTS-Iᴛ happened , especially within each of the development teams, whereas email was the primary means of computer-mediated communication. During the preliminary interview sessions, we asked the participants to quantify the average number of 'internal' emails sent daily and their main purposes. Seven participants (35%) reported sending 10 or more internal emails per day, six (30%) between 5 and 10, and the other seven respondents (35%) less than 5. Regarding the

---

[2]https://automationforjira.com

main purposes for sending internal emails, the most reported reason (13 mentions) was the notification of events (e.g., user story masked completed on Jira, updated a documentation page on Confluence. The other two most cited reasons were *ex aequo* (8) information requests and responses and event organization (e.g., scheduling calls or meetings).

After email usage, we focused on understanding how Slack was being used at KSTS-It. We chose to restrict the analysis of Slack usage statistics to the time-window of Oct. 2018-Jan. 2019 because before then, Slack configuration was different compared to when we started the case study (each project used to have a dedicated channel), but that setup had been already abandoned because deemed dispersive. As such, at the beginning of our case study, we analyzed what public and private channels were available and who had subscribed to them (Table 3). Please, note that we received n=18 answers this time and that the channel mappings for managers M3 and M4 were not collected since they acknowledged not having really used Slack yet at that time.

Regarding the available channels, #core, #devops, and #kdm were used for receiving Jenkins notifications from, respectively, the CORE back-end framework component, the tests executed by the DevOps team, and the deployment of new KDM features. The channels #components-ui and #components-ui-commits were used to collect notifications from GitLab regarding the framework front-end component. The channels #core-internal, #dev-internal, and #devops-internal were the private channels used by the respective team for intra-team communication.

As for the channel subscriptions, the results in Table 3 show that QA members and Managers had subscribed to very few channels. Instead, the DevOps, Core, and Dev team members (in order) had subscribed to more channels, mostly private (shown in grey). All the eighteen respondents reported using the Slack desktop or web app as their daily driver, with only five reporting having the app also installed on their phone. Ten of them (~56%) reported having set the notification preference as '*Direct messages, mentions & keywords*,' with the others equally distributed between '*All new messages*' (22%) and '*Nothing* (22%).'

Regarding the integrations, we found that Slack had been integrated with Jira and GitLab to receive event notifications. During the preliminary and intermediate interviews, we asked about the perceived usefulness of these integrations. Only one participant reported using both integrations, whereas the others had activated either Jenkins or GitLab. They found the stream of notifications from both tools to be overkill because the same notifications would arrive via email.

Next, we analyzed whether and how Slack channels were used. Fig. 2 shows the distribution of messages sent to public channels, private channels, and as direct messages. We notice that, after abandoning the previous configuration, Slack was being used almost exclusively as an IM tool for sharing direct text messages. Otherwise, communication happened via private channels and almost never publicly.

Finally, albeit without access to logs, we directly observed that Skype for Business was also used for text-based, one-to-one chats.
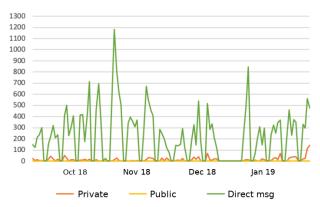

Figure 2: The observed distribution of messages sent over Slack by type (Oct. 2018 to Jan. 2019).

> **Takeaways: Communication-related issues**
>
> Direct observation and interviews revealed that internal communication at KSTS-It was 'unstructured:'
> - Information overload exacerbated by the use of email for internal communication (notifications, info requests & responses, event scheduling).
> - Slack used mostly for internal communication via direct messaging or private channels.
> - Unclear rules about which medium to choose between email, Slack, and Skype for Business.

3.1.2 Workflow Issues. During the interviews, we asked about the SCRUM workflow adopted at KSTS-It. Albeit respondents acknowledged some of the expected and well-known benefits of the Agile methodology (e.g., time-boxed development, rapid client feedback, stand-up meetings, and Sprint retrospectives), there were also some complaints about it as well as related to the extent of tool support.

First, developers and testers complained about the uneven workload that tends to spike at the end of each Sprint release; albeit they acknowledged this as a condition intrinsic to time-boxing, they also thought the problem was partially due to a planning issue, specifically, "*the overly optimistic cost estimation of the user stories in the backlog*" (QA1).

Other issues were mentioned that make the workflow "*not perfect*" (A1). DevOps team members felt that "*the general workflow does not involve us to the right extent*" (DO1) and that "*this often causes misalignment with the other teams, especially with the Dev team*" (DO2).

Second, regarding tool support, while Jira was generally appreciated, there were also complaints about it being confusing at times. After seeking clarification, we found out that the managers and team leaders thought that all the details available in the Jira board were sometimes distracting, for example, when they just want to quickly assess the advancement status of a Sprint, like, which user stories are completed, under test, and to do. Still, they acknowledged that those details are necessary to the development teams nonetheless.

Table 3: Observed mapping of channel subscriptions per interview respondent (n=18). Channels not related to development activity (e.g., #random, #general) are omitted. Rows in grey indicate private channels.

Channels	Dev						QA			DevOps				Architecture			Management	
	D1	D2	D3	D4	D5	TL1	QA1	QA2	TL2	DO1	DO2	DO3	TL3	A1	A2	TL4	M1	M2
#components-ui					x	x									x			
#components-ui-commits						x												
#core		x		x						x			x	x	x	x	x	
#devops		x						x		x	x	x	x		x	x	x	
#kdm										x	x	x	x	x				
#core-internal										x	x	x	x	x	x	x		
#devops-internal										x	x	x	x					
#dev-internal	x	x	x	x	x	x				x	x	x	x	x	x	x		
#techboard	x	x								x					x			

The developers had complaints about Jira too. In particular, they generally reported the need for manually executing actions to synchronize the status of each user story after changing those of its sub-tasks.

Takeaways: Workflow-related issues

Direct observation and interviews revealed the following problems related to the collaboration workflow in Jira:

- SCRUM board filled with details useful for the developers but 'distracting' for managers and team leaders.
- Several, tedious actions needed in the SCRUM board for maintaining up-to-date and in sync the status of issues and their sub-tasks.

3.2 Changes

Building upon the observations and interview outcomes, we implemented the two changes detailed earlier in Sect. 2.3. Here we report our findings.

3.2.1 First Change: Reinforcing the Use of Slack. The focus of the first change was on reinforcing the use of Slack as a central hub for internal communication. To counteract the communication issues that emerged from the interviews and direct observation, during the change, Slack became the tool of choice for channeling all the internal communication. At the same time, the use of Skype for Business and Outlook was recommended for communicating (i.e., email, voice call, screen sharing) with remote sites and customers.

To foster the use of Slack for internal communication, we added to the KSTS-IT workspace three new channels with the following usage rules that aimed at replacing the use of internal emails for development, testing and Sprint organization entirely:

- #dev — a public channel used by the Dev team to report problems, seek advice from others, and share code snippets; this channel replaced the private one #dev-internal;
- #qa — a public channel for hosting internal discussion of the QA team about quality and to notify user story completion;
- #sprints — a public channel for sharing Sprint retrospective reports and urgent communications about the current Sprint, which need immediate managers' attention.

As a result, by using Slack, all the Sprint-related messages were collected in one public place accessible by everyone at KSTS-IT, while also helping to reduce email overload. Please, note that, other than the new three channels, the Slack configuration during the execution of first change also included the previously existing channels reported in Table 3, which we did not analyze.

Table 4: The Coding schema used in this study (from [19]).

Category	Definition	Example
Coordination / General info	Giving general information	*Today IâĂŽm working from home and won't take part in the scrum daily meeting*
General discussion	Q&A regarding general topic	*Is this part of the current Sprint?*
Problem-focused communication	Technical questions and discussions of possible solutions	*Is there anything that we can do to align the two frames?*
Technical information	Giving technical information	*STREAMRSCM-646 closed*
Socializing	Messages used for socializing	*[If you want me to close this task] 10âĆň*
Emoji	Emojis sent by users	*:pray:, :smile:*

Finally, to ensure that all the participants used the tool as intended during the change, we added to Confluence a new documentation page about Slack, with channel descriptions, instructions, and advice for a proper use such as avoid using direct messages and creating new private channels for communication relevant for a whole team.

Quantitative Analysis Results. As already mentioned in Sect. 2.3, data collection for the first change lasted for two weeks (i.e., one entire Sprint) from Feb. 5 to Feb. 19. After collecting the logs of messages exchanged in Slack, we performed both quantitative and qualitative analysis by first counting the number of active users, messages sent, and mentions per user in the new channels; then, we performed the content analysis of the messages.

During the Sprint, 125 messages were sent in the three new channels, distributed as follows: 80 messages in #qa, 32 in #sprints, and 13 in #dev. The #qa channel was also the most active in terms of number of active participants[3] (9) and shared @username mentions (101), followed by #sprints (6 active users and 12 mentions) and #dev (5 and 2).

To understand what kind of information was exchanged in the channels during the Sprint, we manually performed the content analysis of the messages using the coding schema designed by Stray et al. [19] in a similar study. The coding schema is reported for convenience in Table 4.

The coding was performed iteratively by the first two authors. In the first iteration, we randomly selected a sample of 40 thematic units. Each thematic unit [3] was obtained by identifying in the log a set of messages sent by one user, which have the same intent (i.e., are conceptually linked together) even if split into multiple,

---

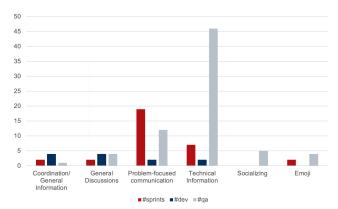[3]An active participant is a team member who shared at least one message.

Figure 3: The results of content analysis with the distribution of categories for the new channels (Sprint Feb. 5-19).

non-consecutive utterances. Otherwise, had we chosen a few random messages and ignored their 'context,' the coding procedure would have been impossible. Furthermore, we selected a number of thematic units from each of the three channels proportional to the overall number of messages sharing therein during the experiment.

The first coding iteration was performed individually, with an inter-rater agreement of 85%; then, the two coders participated in a meeting to resolve disagreements. The second iteration followed a similar process on a new set of 40 thematic units. Because the measured inter-rater agreement was high (90%), after resolving the disagreements, we decided that the remaining units could be safely coded by one coder only. (For simplicity, from now on, we will use the term message.)

Fig. 3 shows the distribution of coded messages across the three new channels. The most common intents for communicating were sharing *Technical Information* (55 messages) and resolving issues (*Problem-focused Communication*, 33 messages). A very few messages were exchanged for general purpose Q&A (*General Discussions*, 10), to coordinate (*Coordination/General Information*, 7), and socialize (*Emoji* 6, *Socializing* 5).

Furthermore, Fig. 3 shows that, overall, the #qa channel was the most active one during the experiment. Also, contrary to expectations, we found a higher percentage of *Technical Information* messages shared in the #sprints channel as compared to #dev, mostly for matters related to releases and execution environment configurations. Albeit communication of this kind is important for Sprint execution (e.g., feature demonstration, Sprint review), this result shows an improper use of the #sprints channel since it was conceived for sharing notification about urgent, blocking issue—to make it more focused, with messages about the previous Sprints always at hand—whereas, technical-related content had to be shared only in the other two channels. Relatedly, despite the small difference, KSTS-It team members shared more messages coded as *Coordination/General Information* in #dev rather than in #sprints, a further proof of improper use of the latter channel.

Finally, regarding the few messages coded as *Socializing* and *Emoji*, we note that the use of Slack at KSTS-It is intended exclusively for work-related matters. Therefore, it is likely that socialization happens through other channels (e.g.,) not considered in this case study.
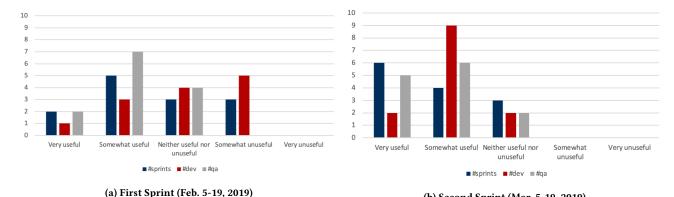
Qualitative Analysis Results. At the end of the Sprint, we conducted semi-structured interviews with 13 participants (the entire Dev and QA teams, TL3, TL4, M1, and M2) to get feedback about how the use of Slack affected internal communication at KSTS-It. First, we asked whether they happened to send emails and, if so, why. We found out that only 4 out of 13 fully complied with the instructions and used Slack instead of using of 'internal' emails. The reasons that they mentioned included: (i) *privacy concerns*, e.g., one developer did not want to negotiate time-off with a manager on a public channel, so they chose email instead of private messaging via Slack; (ii) *notification overload*, i.e., one developer reported a bug found in the Core framework via email instead of using #core as established because the channel became overloaded by the continuous flow of Jenkins notifications, which ended up making any human interaction impossible; (iii) *IM replacement*, i.e., users had the (wrong) perception that Slack was better suited for short, instant messages and, hence, the use of emails was preferred for longer, more complex communication like discussing a feature from the current Sprint.

Fig. 4a reports the results of a survey for measuring the perceived level of usefulness for the three new channels (#dev, #qa, and #sprints) on a 5-point Likert scale, anchored with the values 1=*Very unuseful*, 2=*Somewhat unuseful*, 3=*Neither useful, nor unuseful*, 4=*Somewhat useful*, and 5=*Very useful*. Before commenting on the results, we point out that, during the follow-up interviews, two managers and the QA team leader acknowledged that they always chose the mid-point category. They told us that they felt unable to fully evaluate the usefulness of the solution because still not used to Slack and also because some of these new channels were not a big part of their workflow.

Regarding the #dev channel (median = 3), 5 out of 13 respondents reported finding it *Somewhat unuseful*. When asked to elaborate during the follow-up interviews, they told us that because in the Dev team "*[they] are lucky to be co-located in the same open-space office*" (D1), is the preferred option for intra-team communication at KSTS-It. Also, they seldom share a problem or ask for feedback team-wide (i.e., in the public channel); instead, they rather need communication tools to "*exchange direct messages during pair programming sessions*" (D2).

The #qa channel was the most appreciated (median = 4), as 9 respondents found it *Very* or *Somewhat useful*. The main reason for the appreciation was that all the email notifications of completed user stories were replaced by messages in the public #qa channels. QA team members found those emails to be "*dispersive, kind of wasted for a one-sentence-only message*" (QA2), whereas with Slack notifications "*things are much faster and more immediate than emails*" (D3). In general, the QA team felt that with Slack, their workflow had become "*simpler, a bit more streamlined*" (QA2).

With respect to #sprints (median = 4), the evaluation was slightly less positive than #qa, with 7 respondents finding it *Very* or *Somewhat useful*, and 3 who thought it was *Somewhat unuseful*. When interviewed, the participants said that they saw potential in it but also that the channel would "*take more than a Sprint to come to fully understand its benefits*" and it would likely become "*more useful over time*" (D1). Therefore, we decided to collect survey responses from the same participants again, after another Sprint (i.e., two more weeks, see Fig. 4b). Compared to the results from the previous

(a) First Sprint (Feb. 5-19, 2019)

(b) Second Sprint (Mar. 5-19, 2019)

Figure 4: Perceived levels of usefulness of the three new channels (a) after the first Sprint and (b) after the second.
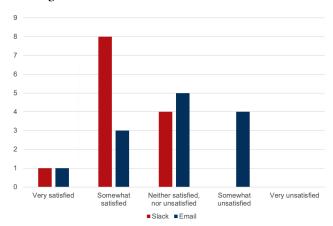

Figure 5: Perceived levels of satisfaction: email vs. Slack.

Sprint, we notice that an increase in the level of perceived usefulness of the #dev channel in the second Sprint as compared to the first one (median 4 and 3, respectively). No noticeable differences are present regarding the other two channels, instead.

Finally, during the follow-up interviews, we collected the perceived levels of satisfaction with using email internally as compared to Slack. Fig. 5 shows that the participants were more favorable to Slack (median = 4, *Somewhat satisfied*) than email (median = 3, *Neither satisfied, nor unsatisfied*), with no one feeling unsatisfied about the former.

3.2.2 Second Change: Multiple Scrum Boards and Jira Automation Rules. In the second change, we focused on improving the collaboration workflow in Jira. To counteract the workflow-related issues that emerged from direct observation and interviews, we first created a new, higher-level Scrum board meant in particular for the managers and team leaders. This new board shows only the main user stories, tasks, and bug reports, and provides at a glance a picture of how the current Sprint is going, with the original board showing the detailed sub-tasks useful for the Dev and QA teams. Furthermore, the high-level board features two new intermediate states (columns): *Ready to test*, for any issue not yet taken care of by QA, and *Testing*, for issues currently under test. Therefore, during the second change, the issues in the new board transitioned into

the following states: *To do*, *In progress*, *Ready to test*, *Testing*, and *Done*.

The other proposed improvement was the installation of *Automation for Jira*, a third-party Slack plugin that allows the definition of automation rules in the form *event–condition–action*. Examples of events are the creation and modification of issues; conditions can be expressed via JQL (Jira Query Language), regular expressions, and graphically through the plugin; typical actions are sending emails and Slack messages through webhooks. For the sake of space, next, we report only a few examples of the ten automation rules defined for user stories while omitting those for bugs and tasks, which are nonetheless similar; we also point out that these rules have been activated for projects developed only at the Italian site and not shared with Berlin: acmcopyright

- `sprint_started`: when a new Sprint is marked as *Started* in Jira, send a notification to #sprints;
- `start_implementation`: when the state of the first sub-task changes from *Open* to *Implementation*, move the parent user story from *To do* to *In progress*;
- `ready_to_test`: when all the sub-tasks are marked as *Closed* (completed), move the parent user story into *Ready to Test* and notify the testers in #qa;
- `all_done`: when all the sub-tasks are closed and tested, the parent user story is marked as *Closed* and a notification of user story completed is sent to #sprints;
- `version_released`: when a new version of project in Jira is marked as *Released*, send a notification to #sprints.

The expected benefits of the second change were: (i) declutter the Scrum board for managers and team leaders and, by separating completed stories from those under testing, allow them to identify possible bottlenecks if user stories pile up in the *Ready to Test* column; (ii) relieve the development teams from the burden of keeping Jira and the others up-to-date after every change.

Quantitative Analysis Results. The data collection for the second change lasted for four weeks (two Sprints) from Feb. 19 to Mar. 19, during which 932 messages were published in the five active channels. Also in this case, we performed the content analysis of the messages using the same coding schema used before (see Fig. 6).

The #sprints channel received 107 messages and 18 mentions from 16 active users. Compared to the first observation, there was
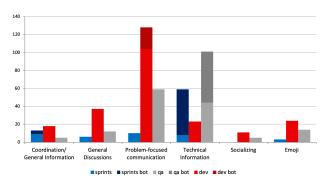
Figure 6: Classification of messages exchanged between Feb. 19-Mar. 19 (plugin notifications shown on top of each bar, if any).

reduction in *Problem-focused communication* (see Fig. 3 vs. Fig. 6); at the same time, there was an increase of the *Technical Information* category, although most of the messages therein (86%) are from the Jira automation plugin. The #dev channel received 267 messages and 87 mentions from 17 users, with an overall in increase compared to the first observation. This increase was caused mainly by having part of the Dev team working remotely during these Sprints and also by the automatic notifications of defects found by QA (~83% of total messages).

Regarding the #qa channel, it received 218 messages and 173 mentions from 15 users. Most of these messages (~56%) are notifications from the Jira automation plugin, such as 'user story ready to test' or 'bug resolved' with explicit mentions of some QA team members. The #core channel was seldom used during the two Sprints (24 messages and 2 mentions from 8 users) and mostly for *Problem-focused Communication*. The #devops channel received 316 messages and 54 mention from 19 users. Most of the messages are coded as *Problem-focused communication* whereas *Technical Information* messages were mostly (60%) automatic notifications from GitLab.

Finally, regarding the two most common categories, we found that, overall, most of the messages classified as *Problem-focused Communication* are from users rather than plugin notifications (284 vs. 24) whereas the opposite is true for *Technical information* (85 vs. 120). This difference is statistically significant ($\chi^2 = 156.97$, $p < 0.01$), meaning that people at KSTS-It interact over Slack mostly for problem-solving.

Qualitative Analysis Results. At the end of the Sprints, we interviewed all the twenty participants. From these follow-up interviews, we found that the high-level board was not used and appreciated by Management to the expected extent. Contrary to expectations, it was the Dev team to find it useful to get a "*bird's-eye view*" (TL1) and an understanding of "*user stories' progress in the second half of a Sprint and know how many are still under development and to do, which is a critical piece of information to us*" (D1). Instead, one manager (M3) admitted to having forgotten about it; still, he added that he found the idea promising and intended to try it during the next Sprints. Another manager (M4) did not appreciate the lack of distinction "*between user stories for which development has not started yet and those that went back into 'To do' because of defects found.*" He, therefore, suggested adding a new

Table 5: Perceived usefulness of automation rules (1=most useful, 10=least useful).

	Dev	QA	Managers	Overall
1	ready_to_test	ready_to_test	start_implement.	ready_to_test
2	defect_found	start_implement.	defect_resolved	defect_found
3	defect_resolved	defect_resolved	defect_found	start_implement.
4	start_implement.	sprint_started	all_done	defect_resolved
5	sprint_started	defect_found	sprint_started	sprint_started
6	default_subtasks	default_subtasks	testing	default_subtasks
7	testing	version_released	ready_to_test	testing
8	version_released	sprint_completed	sprint_completed	version_released
9	sprint_completed	testing	default_subtasks	sprint_completed
10	all_done	all_done	version_released	all_done

column *Defect resolving* to handle the latter case. He also suggested adding a feature to quickly filter issues based, e.g, on the assignee, reporter, which would be useful during daily meetings and Sprint retrospectives.

Furthermore, we asked the participants about the perceived usefulness of the *Automation for Jira* plugin. Four respondents found it *Very useful* and seven *Somewhat useful*, with only two managers finding it *Neither useful, nor unuseful*. We also asked them to rank the ten automation rule in descending order of usefulness. The results are reported in Table 5 aggregated per team and overall. The top three ranked rules are ready_to_test ("*[it] saves a lot of time not to have to notify testers when a new story is ready for them*", D1), start_implementation ("*avoiding manual changes in Jira so convenient*", D4), and ("*interaction is more direct and immediate*", QA1). While the ranks of the Dev and QA teams are very similar to each other, there are some differences with the managers, who are more interested not only in being notified about defects being found and fixed but also about the beginning and end of implementation.

Thanks to the plugin, 85% of the interviewees found their workflow to be improved, and the remaining found it unaffected by it. No participant found it to have a negative effect ("*automation is always good*", TL1), and all agreed to keep on using it afterward.

Finally, we asked the interviewees about the perceived usefulness of the channels #core and #devops that were not analyzed during the first change. Thanks to the feedback collected from the first Sprint, we applied some modifications in the policies of their use. First, we disabled the automatic notifications from Jenkins, which were deemed uninteresting *per se* and were so numerous that they prevented any human interaction. Also, all participants were recommended to set their preferences to *Direct messages, mentions & keywords* not to miss any relevant notification. Regarding #core, most of the interviewees (8) found it *Neither useful, nor unuseful*. Architecture team members A1 and TL4 agreed that this is because "*[we] rather chat because architectural discussions are too long to type in*" (A1). The #devops channel was better appreciated by the interviewees (4 *Very useful* and 4 *Somewhat useful*). Without Jenkins notifications plaguing it, the channel was found to be "*more focused, with only relevant info*" (DO1), as it allowed them to "*keep in touch with the Dev team and help them fix configuration issues*" (DO2). Also, they appreciated that it is useful "*not only to DevOps but everyone, being able to see when a request is made, it helps increase the awareness in the teams overall*" (DO3).

4 DISCUSSION

In this section, we first discuss the findings from our case study on how to improve tool support to collaboration at KSTS-IT. Then, we report the implications of our findings and discuss their limitations.

4.1 Slack as a Central Hub of Communication

After observing inconsistencies in the way communication tools were used at KSTS-IT, as a first change, we reinforced the use of Slack as the central hub for internal communication, while setting clear rules regarding other tools and channel usage.

Slack fosters communication transparency. Thanks to the use of public channels for sharing team-relevant pieces of information, the new Slack configuration introduced helped to make communication at KSTS-IT more 'transparent.' As a further confirmation of this finding, we compared the Slack analytics from two months before and after the case study and noticed that the number of messages sent to public channels increased from 0.1% to 4.4% while, at the same time, direct messages decreased from 92% to 85% and those sent to private channels went down to 7.9% from 10.6%. This increase in public messages was the result of both ensuring everyone complained with the recommended notification setting (i.e., no more messages went unnoticed) and managers getting more and more comfortable with Slack.

Slack is for everyone (to different extents). Overall, KSTS-IT teams liked using Slack as a central hub of communication and used it mostly problem-solving. However, teams appreciated it to different extents. QA and DevOps teams were the most favorable, although it also grew on developers during the next Sprints, whereas the solution turned out to be not as suitable for the Architecture team members who found that their long, intra-team discussions and brainstorming sessions about the CORE framework are better served .

4.2 Jira Boards and Automation Rules

As a second change, we added to Jira a new high-level SCRUM board while also introducing automation rules and notifications to Slack.

Managers need more time to adjust. The feedback interviews revealed that the high-level Jira board was not used by managers as expected—development teams seamed to appreciate it better—yet they told us they wanted to keep trying to use it. We observed similar behavior in the case of Slack. Therefore, our experience suggests that managers may need more time to adjust and get used to changes in the way collaboration tools support their workflow.

The more (automation) the merrier. Overall, the ten automation rules introduced in Jira were found useful, with 85% of participants reporting a more streamlined workflow thanks to them. Besides, while automation rules may not be as useful for everyone, they are detrimental to no one. As such, KSTS-IT is considering expanding the set of automation rules also to other tools.

4.3 Related Work

Beware of automatic notifications from development tools. Calefato and Lanubile [5] proposed a Hub-and-Spoke model to loosely integrate software development and collaboration tools, and create a central hub to fight channel fragmentation and communication overload. They also implemented a prototype that used Slack as a central hub. In our case study, the first change restructured the existing Slack workspace configuration in a way that broadly follows the model recommended in [5]. Our findings suggest that while the model helped to make pieces information otherwise fragmented all available in one place, there is also a severe risk of overload of automatic notifications from development tools connected to Slack, which can even hinder human interaction.

Use separate, public channel for each team. Stray et al. [19] investigated the use of Slack for coordination in a distributed software organization. While we focused on the intra-site communication of a global software company, we reused their coding schema to understand what was discussed on Slack at KSTS-IT. Stray et al. observed mostly *Problem-focused Communication* (48%), followed by *Technical Information* (20%) and *General Discussion* (15%). If we ignore the automatic notifications, we consistently find that most of the messages shared during the case study belong to the *Problem-focused Communication*. Despite these differences in communication intent, we notice that Stray et al. created a set of final guidelines that closely resembles the configuration of the Slack workspace in our case study, including (i) use public channels to foster open communication, (ii) create one channel per team, and (iii) aggregate and archive discussions in one place to make it searchable.

4.4 Limitations

As with every study, there are some limitations to our work. First, we followed a longitudinal, single-case design and, therefore, the general criticism of uniqueness applies. Another limitation related to generalizability is that we were allowed to analyze collaboration and communication only within the Italian site (quite limited in size), rather than between sites, where interaction also happened F2F. A second limitation relates to potential bias in our data collection. Some data was based on direct observations and semistructured interviews conducted by one of the authors. However, to reduce bias, multiple data sources were used to triangulate relevant pieces of information, and a protocol was agreed upon before conducting the interviews. Finally, another limitation is that we did not analyze the effects of the implemented changes on communication happening through the other channels, such as F2F and email.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we reported on a case study conducted at KSTS-IT, the Italian site of a large, distributed software company. We analyzed their Agile tooling and work environment, and proposed a couple of changes in the way they used Slack and Jira to improve the collaboration. We found out that, overall, KSTS-IT accepted the restructured Slack workspace and the Jira automation rules, and reported more improvements in their communication and workflow.

As future work, KSTS-IT is investigating the idea of extending the role of Slack as a central communication hub by adding more integrations with other tools, such as Confluence, and expanding the set of automation rules in Jira. Finally, they are considering also bringing to Berlin some of the changes investigated at the Italian site.

REFERENCES

[1] David E. Avison, Francis Lau, Michael D. Myers, and Peter Axel Nielsen. 1999. Action Research. *Commun. ACM* 42, 1 (Jan. 1999), 94âĂŞ97. https://doi.org/10.1145/291469.291479

[2] Barry Boehm. 2006. A View of 20th and 21st Century Software Engineering. In *Proceedings of the 28th International Conference on Software Engineering (ICSE âĂŹ06)*. Association for Computing Machinery, New York, NY, USA, 12âĂŞ29. https://doi.org/10.1145/1134285.1134288

[3] Fabio Calefato, Daniela Damian, and Filippo Lanubile. 2012. Computer-mediated communication to support distributed requirements elicitations and negotiations tasks. *Empirical Software Engineering* 17, 6 (2012), 640–674.

[4] Fabio Calefato and Filippo Lanubile. 2011. A planning poker tool for supporting collaborative estimation in distributed agile development. In *Proc. 6th Int. Conf. Softw. Eng. Adv.(ICSEA)*. 14–19.

[5] Fabio Calefato and Filippo Lanubile. 2016. A Hub-and-Spoke Model for Tool Integration in Distributed Development. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. 129–133. https://doi.org/10.1109/ICGSE.2016.12

[6] Mike Cohn. 2005. *Agile estimating and planning*. Pearson Education.

[7] Paulo SÃrgio Medeiros dos Santos and Guilherme Horta Travassos. 2011. Chapter 5 - Action Research Can Swing the Balance in Experimental Software Engineering. Advances in Computers, Vol. 83. Elsevier, 205 – 276. https://doi.org/10.1016/B978-0-12-385510-7.00005-9

[8] Marilyn M Helms and Judy Nixon. 2010. Exploring SWOT analysis–where are we now? *Journal of strategy and management* (2010).

[9] SF Lee and Andrew Sai On Ko. 2000. Building balanced scorecard with SWOT analysis, and implementing âĂIJSun TzuâĂŹs The Art of Business Management StrategiesâĂİ on QFD methodology. *Managerial Auditing Journal* 15, 1/2 (2000), 68–76.

[10] Pernille Lous, Marco Kuhrmann, and Paolo Tell. 2017. Is scrum fit for global software engineering?. In *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*. IEEE, 1–10.

[11] Pernille Lous, Paolo Tell, Christian Bo Michelsen, Yvonne Dittrich, and Allan Ebdrup. 2018. From Scrum to Agile: A Journey to Tackle the Challenges of Distributed Development in an Agile Team. In *Proc. of 2018 Int'l Conf. on Software and System Process (ICSSP '18)*. ACM, New York, NY, USA, 11–20. https://doi.org/10.1145/3202710.3203149

[12] Pernille Lous, Paolo Tell, Christian Bo Michelsen, Yvonne Dittrich, Marco Kuhrmann, and Allan Ebdrup. 2018. Virtual by Design: How a Work Environment Can Support Agile Distributed Software Development. In *Proc. of 13th Int'l Conf. on Global Software Engineering (ICGSE '18)*. ACM, New York, NY, USA, 102–111. https://doi.org/10.1145/3196369.3196374

[13] G. Murphy. 2009. Attacking information overload in software development. In *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 4–4. https://doi.org/10.1109/VLHCC.2009.5295312

[14] Efi Papatheocharous and Andreas S Andreou. 2013. Evidence of agile adoption in software organizations: An empirical survey. In *European Conference on Software Process Improvement*. Springer, 237–246.

[15] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14, 2 (2009), 131.

[16] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.

[17] Ken Schwaber and Mike Beedle. 2002. *Agile software development with Scrum*. Vol. 1. Prentice Hall Upper Saddle River.

[18] M. Storey, A. Zagalsky, F. F. Filho, L. Singer, and D. M. German. 2017. How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. *IEEE Transactions on Software Engineering* 43, 2 (Feb 2017), 185–204. https://doi.org/10.1109/TSE.2016.2584053

[19] Viktoria Stray, Nils Brede Moe, and Mehdi Noroozi. 2019. Slack Me if You Can!: Using Enterprise Social Networking Tools in Virtual Agile Teams. In *Proc. of 14th Int'l Conf. on Global Software Engineering (ICGSE '19)*. IEEE Press, Piscataway, NJ, USA, 101–111. https://doi.org/10.1109/ICGSE.2019.00031