Software Development and CSCW: Standardization and Flexibility in Large-Scale Agile Development

HELENA TENDEDEZ, Lancaster University, Lancaster, UK
MARIA-ANGELA FERRARIO, Lancaster University, Lancaster, UK
JON WHITTLE, Monash University, Melbourne, Australia

Identifying which agile methods and processes are most effective depends on the goals and aims of an organisation. Agile development promotes an environment of continuous improvement and trust within self-organising teams. Therefore, it is important to allow teams to have the flexibility to customize and tailor their chosen methods. However, in a large-scale agile deployment, there needs to be a degree of process standardization across the organisation; otherwise, different teams will not be able to effectively share knowledge and best practices. This paper addresses this classic CSCW issue of the tensions that arise between process standardization and flexibility in a large-scale agile deployment at the BBC.

CCS Concepts: • **Human-centered computing~Empirical studies in collaborative and social computing** • **Software and its engineering~Software creation and management**

KEYWORDS

Agile development, software development, empirical study, flexibility, standardization

1 INTRODUCTION: Software development and CSCW

Software development has featured strongly within CSCW research as an example of cooperative work. For example, Schmidt and Sharrock [40], Dittrich [39], Button and Sharrock [41]; Grinter [42] all point to the importance of coordination, the use of representations and tools, as well as organizational obstacles and constraints. In recent years agile development has been highlighted as an approach that relies on what some might regard as classic CSCW properties in the use of a repertoire of coordination mechanisms, communication modes and tools, artefacts and structuring devices while following an agile, user-centered development approach [43]. Agile development is a software development methodology that is responsive to changing requirements. It follows a set of principles aimed at building software in a flexible environment, through collaboration of self-organizing, cross-functional teams [1]. Organizations can adopt an agile methodology wholesale, or implement different tools or processes to conform to the agile development principles. However, organizations will continually evolve and tailor their agile development processes to suit the needs of the organization [3]; that is; any application of agile development in an organization is a practical interpretation of the agile principles outlined in the Agile Manifesto [1]. Interpretations of how to practice agile development will differ between organizations and agile teams [3]. Where there is opportunity for interpretation of these principles, there is freedom to select and adopt practices that are most effective and valuable in a particular environment [3, 4, 27]. There is added challenge for large-scale organizations which apply agile methodologies, due to reasons such as difficulties of coordinating large development teams, preserving productivity and dealing with large and complex software [5, 10, 21]. As agile development is relative to each organization, different organizations will require different agile scaling strategies and techniques to remain effective [2]. Large-scale organizations must consider if their interpretations of agile development work at scale, as at scale, strategies can change and restrict flexibility [2, 4, 5, 6, 27]. For example, standardizing agile processes through established frameworks may make process alignment across teams simpler, and best practice sharing more consistent [27]. However, agile methods are organization-dependent, hence it has been suggested that selecting a few methods may be more suitable than adapting wholesale to a single agile framework [28]. This is because frameworks can limit flexibility typically encouraged by agile development values [4, 6, 24]. Larman and Vodde [24] discuss the

difference between a self-managing agile team's genuine desire to hold a daily stand-up meeting and holding a stand-up meeting to conform to an imaginary 'we are doing agile' checklist.

This paper explores the tensions between standardization and flexibility in a large-scale agile environment. *Standardization* refers to agile processes that are enforced as a standard protocol across an organization to share knowledge and best practice. *Flexibility* refers to the ability to customize and evolve processes to suit the aims of an agile team. Tensions occur when the desire to standardize certain processes prevents the ability to be flexible, whether this occurs within agile teams, from team to team or externally. We contribute an exploration of these tensions, and discuss the challenges of balancing flexibility and standardization in large-scale organizations as informed by our findings. This challenge can be seen as an extension of the 'classic' CSCW problem, 'what to automate and what to leave to human skill and ingenuity'.

2 RELATED WORK

2.1 Large-Scale Agile Development

Large-scale agile organizations can be defined by the number of software teams within the organization, the lines of code for a solution, or the cost of the agile project [7]. Determining scale based on number of teams suggests that 2-9 software development teams is large-scale, with 10 or more teams being 'very large' [7]. Scaling agile development usually refers to increasing the number of teams or their sizes, or scaling in respect to geographical location [6]. Scaling team practices, managing multi-site organizations and distributed teams are among some of the challenges when scaling agile development [8, 9]. It is thought that large-scale agile development is less effective, with levels of agility being limited by scaled environments [14]. Knowledge sharing across teams in a large-scale agile organization is an example of the challenges faced with agile at scale [35]. Understanding agile development and large-scale projects, followed by what effects self-organizing teams was among the 'Top 10 Burning Questions from Practitioners' in 2010 [11]. Five years later, scaling agile development and self-organizing teams was once again noted as a key challenge in practice, highlighting the relevancy of the ongoing problem today [9]. Allowing self-organizing teams to independently decide how to reach organizational goals versus employing organization-wide standardized processes that each team must follow is an example of problems arising from scaling strategies [24]. Experimenting with practices and facilitating novel practices were noted as key guidelines for tailoring agile in the large by Rolland et al. [36].

2.2 Coordination and Interaction

Coordination and interaction are central to CSCW; indeed, as Schmidt [61] notes, CSCW can be said to have been born with the concerns surrounding how various cooperative groups can be better enabled to coordinate their interactions to ensure reliability and efficiency. Standardized processes across agile teams can be beneficial, such as daily stand up meetings, which can improve communication and promote team awareness [17]. Standardizing processes can also help with alignment and coordination in large-scale environments [27]. However, the daily demand to remain socially active in agile environments can be stressful for team members [17]. Dourish and Belloti [31] discuss different approaches to awareness and coordination in successful collaborative environments. Particularly, they note that there is a cost associated in collaborative working environments when individuals must exchange and produce information through restrictive structured processes [31]. In distributed teams, lightweight mechanisms to support awareness and information sharing can help build a sense of community and shared understanding of work [32], which can be difficult with visual information sharing in agile environments [33].

Alongside one of twelve agile principles being to efficiently and effectively '[convey] information to and within a development team' through 'face-to-face conversation' [1], it has been highlighted in CSCW literature that emotional context can be lost when replacing face-to-face

communication with email and other communication tools [34]. As larger and distributed teams collaborate, there is a shift towards more formal, controlled and well-defined interaction processes through computer-supported methods [34]. Schmidt and Simone [36] describe the importance of being able to change the configuration of organizational-wide protocols (such as Kanban systems) in order to adapt to changes in organizational demand and requirements. Adapting such processes to suit new situations requires direct and informal cooperative arrangements between team members in order to increase the flexibility of existing systems [36]. This requires that team members can take control of existing protocols and systems in order to adapt them and keep them flexible when there is a change in demand [36]. In some instances, the adaption methods can lead to processes going against core agile beliefs. Hoda et al. [51] describe how extensive documentation is necessary for project communication between teams and customers, and cannot be avoided as encouraged by the Agile Manifesto [51]. For example, software used for airline cockpits requires extensive documentation for certification processes [51].

2.3 Flexible Processes and Standardization

One of the main attraction points of agile development is the flexibility offered and the positive impact that this flexibility has on development activities [15]. Empowering self-directed teams to exercise flexibility increases levels of trust and respect for one another, and can improve performance [16, 27]. Agile development processes are by nature transient and can lose effectiveness as time progresses [12, 13]. Therefore, standardization of processes can undermine the values of agile software development, as processes should be open to adaption and experimentation [27]. This is where an organization may come across tensions between standardizing practices and allowing flexibility of processes in and between agile teams. Hoda et al. [51] discuss the benefits of adapting agile methods to suit different projects and contrast this with the view that agile should be done 'by-the-book'. They note that the 'but Agile says…' by-the-book mindset is unhelpful in understanding the best methods to employ for a particular project.

The autonomy provided by agile development to teams requires a change in mindset, which can be a difficult and slow process [53]. The boundaries between the project management and development roles become blurred when there is an expectation for all team members to possess some level of autonomy and heightened responsibility (contrasting with a traditional software development environment) [53]. However, the ability to continually adapt and take ownership of different agile methods has shown to have a positive impact on team members during development activities [15, 27]. This allows for team members to innovate with their development process, with the ability to craft processes to create an optimal development cycle. Standardizing software development processes also allows for well-established, software best practices to be followed by organizations, such as through software capability maturity models [18]. Standardization may also make interactions and relationships required for metawork more uniform and consistent, but can reduce choice and in turn, flexibility [34]. Typical hierarchical decision-making structures within an organization can cause resistance from agile teams [15, 27]. Identifying the best way to balance flexibility with the top-down impositions of standardizing a single interpretation of agile development across an organization is an area with limited exploration. Once this area is understood, contributions can be made to the maintenance of successfully aligned large-scale agile environments that retain empowerment and flexibility to self-organizing teams.

3 METHODOLOGY AND CONTEXT

Dingsøyr et al. [19] have noted the slow increase of empirical research surrounding agile development, creating a barrier between research and practice. They note that studies need to be based on empirical data rather than argument alone to progress, with challenges being overcome through theories built and tested within industry [19, 20]. We expand on the existing body of empirical work carried out in software engineering contexts [47, 48, 49, 50, 51, 52, 54] through our case study presented in this paper, demonstrating how specific tensions arise when attempting to balance flexibility and standardization in a large-scale agile environment.

The motivation for this study was to assess the ways in which standardization impacts on flexibility in a large-scale agile environment [7]. We achieved this through the ethnographic interview and observation of the culture and agile practices within teams at the BBC's Television and Mobile Platform (TVaMP) at Media City UK, which had 16 agile teams at the time of the study. This continues the ethnographic turn in CSCW, which developed out of an interest in documenting the details of work arrangements; a sophisticated view of the relationship between work and technology (Suchman [45]; Hughes et al [46]; Heath and Luff [38]); and a concern to understand a social setting as perceived by its participants (or users). Our study is comprised of 17 semi-structured interviews and 9 days of disclosed direct observation [23], carried out at the BBC TVaMP in MediaCityUK. This choice in methodology allowed for the capture of a rich qualitative dataset, without interfering with the day-to-day working schedule of the employees. The interviews were designed to discover interesting initial insights and understand the research context, which were then followed up during the observation. The observation provided depth to the interview data, whilst providing the ability to observe and speak to the agile teams in their natural work setting.

3.1 Interviews

Semi-structured interviews were conducted as the first means of data collection for the study. A total of 17 interviews, each lasting 30-60 minutes were carried out over 5 weeks at the site. A semi-structured approach allowed each participant the freedom to express their views on a range of topics without being constrained by any fixed, systematic questions. A mixture of open and closed questions were asked. This was an important approach for our emergent methodology, as we wanted the initial interviews to uncover insights that were not geared towards specific pre-defined topics. Interview questions were updated around every 2nd to 3rd interview to explore interesting emerging themes and ideas. We asked questions about what agile methods were used to manage software projects at TVaMP, how agile methods were mixed and matched, what they believed were the benefits and challenges of the agile environment and how teams self-organized.

Interview participants were recruited by the commissioner of the study at the BBC, with the aim of recruiting a diverse range of participants (in terms of job titles and number of years at the corporation). Potential participants were approached via an e-mail sent by the commissioner of the study. Those who expressed interest in discussing their thoughts on agile development at their department were invited to participate. Table 1 summarizes the range of participants interviewed.

Table 1. The roles of interviewed participants and their experience with software development

Participant's Job Role	Experience with agile development (years)	Experience with software development (years)
Software Engineer 1	< 1	< 1
Software Engineer 2	2+	7+
Software Engineer 3	3+	8+
Software Engineer 4	5+	9+
Software Engineer 5	< 1	< 1
Portfolio Project Manager	-	-
Project Manager	1+	5+
Technical Project Manager	3+	7+
Technical Project Manager	5+	9+
Software Tester	-	-
Senior Test Engineer	2+	3+
Business Analyst 1	3+	8+
Business Analyst 2	5+	5+
Technical Lead	7+	10+
Development Team Lead	3+	5+

3.2 Direct Observation

A disclosed direct observation at the BBC site in MediaCityUK was the second data collection method. Direct observation involves unobtrusively observing environments as they occur, to not bias the observations [23]. This method was the most appropriate given the study's exploratory nature. As participants knew they were being observed, the type of observation was disclosed observation. A 9 day observation lasting 6 hours per day took place at the site (54 hours in total). Data was gathered through being embedded into different teams, sitting in on meetings, discussions, forums, and pair programming sessions. We also had the opportunity to briefly visit other agile departments in the BBC building. The activities chosen for observation were mainly decided based on the emerging themes from interviews. For example, as communication was a recurring theme from the interviews, activities centered on communication such as stand up meetings were observed. However, there was also a degree of spontaneity when choosing activities to observe, depending on the types of activities that were taking place each day. Detailed field notes were taken during each day of the observation. At the end of each day, emerging data was mapped with previously collected data (through the interviews and the previous days of observation). Key themes which developed during the first days of the observation were used to fuel the questions asked and activities observed for the following days. In total, we observed 12 daily stand ups, 9 pair programming sessions, 4 retrospective meetings, 4 software testing sessions, 9 meetings and 1 demonstration or 'demo'.

3.3 Analytical Approach

Interview and observational data were analyzed using a thematic analysis approach [22]. The intention with this study is not to extract theory from the empirical observations, but to understand how things work in the specifics of practice. Coding of the interview transcripts and observation notes was carried out by hand by the first author, which was then iteratively cross referenced and discussed with author three. Thematic analysis was the most appropriate method of analysis given the nature of the collected data. The interview transcripts and observational notes were carefully analyzed at a sentence level, coded into themes and then grouped into larger parent themes. Both datasets were then compared and any relationships between the themes in the data were identified and discussed among the authors, forming a taxonomy of the data. Themes which emerged from the data were grouped under the parent themes: process, organizational or social factors. Table 2 summarizes the taxonomy of themes and sub themes.

Table 2. The themes of the interview and observation data.

Parent theme	Definition	Sub themes
Process	Processes explore specific agile processes used within the crews and the organization as a whole. This includes how processes are defined, used and their perceived usefulness.	Process Experimentation, Flow, Problems with Standardized Processes, Challenges with Scrum, Challenges with Kanban
Organizational	Organizational factors explore the way that the organization manages crews, organizes itself internally and aligns with external stakeholders.	Coping with Scale, Crew Factors, Dependencies, Culture, Top Level Authority Perceptions
Social factors	Social factors relate to the way crews interact and make decisions.	Trust, Communication, Crew Expectations

3.4 Context

At the time of the study, the BBC in MediaCityUK was made up of many departments that practiced different software development methodologies. Some used an agile approach and some did not.

This study focuses on TVaMP, who operated across a single open-plan floor. Within the department, there were 16 teams, or crews as they are called at TVaMP, ranging in size from approximately 6-10 members. This made the department a very large-scale agile environment [7].

3.4.1 Platform and the Crews. There were 7 main groups in the platform which crews worked within, split by product or service. These were: *Red Button* (a digital interactive television service), *Sport*, *iPlayer* (an online video streaming service), *Mobile Core Engineering* (dealing with the iOS and Android mobile features and experience), *Certification Team* (dedicating to testing devices in order to ensure they are compatible with *iPlayer* and *Red Button*), *UX Design* (consisting of user experience designers that join specific crews where they are needed) and *Platform Health* (dedicated to working on tickets that have been raised as containing bugs or minor problems by other crews). Crews within the broader groups were occasionally dependent on one another, and during periods of high demand on the service, crews from different groups would depend on one another (for example, during the season of a high profile sporting event, *Sport* and *Red Button* and *Mobile Core Engineering* may all depend on one another). Crews may often have a weekly dependency on *Platform Health,* while minor issues and bugs are resolved.

A typical crew was made up of 4 developers, 2 testers, 1 business analyst (BA) and 1 project manager (PM) who worked collaboratively. Interview participant 1, a PM, described their unconventional role in the agile crew as "My role can do anything to helping out with the testing, requirements definitions, to helping out managing expectations internally and externally, breaking down problems … if you looked at a traditional PM for an external organization, it would be very much managing budgets, creating Gantt charts and might say 'we are 62.8% complete' – that means nothing. So I say my roles and responsibilities is what I will move whenever there is a blockage in the workflow, to fundamentally get that value out that we are working on".

The number of members or roles within a crew varied and members were occasionally rotated between crews for skill building. Crew members were managed by their respective PMs. However, each technical role (tester or developer) was additionally managed by a technical lead. One technical lead was responsible for all the testers and a separate lead for the developers. Technical leads had responsibilities such as overseeing the technical work of the platform and running workshops for those in technical roles to learn about new tools and technologies. Technical leads could be developers from regular crews. However, most their time was spent managing testers or developers, rather than being embedded in a specific crew. Fig. 1. depicts the relationship between crews and technical leads.

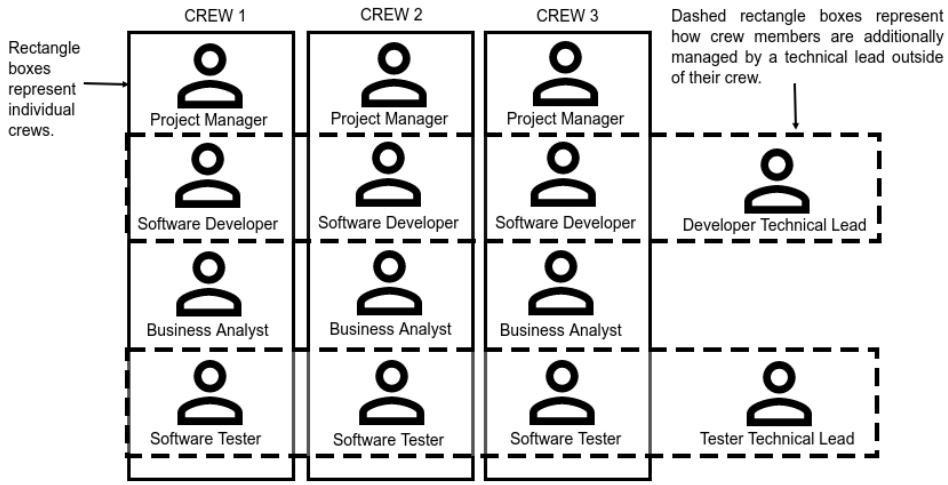

Fig. 1. How testers and developers within crews are additionally managed by technical leads.

Despite the platform being divided into different groups, the platform had a shared goal of providing a heightened digital broadcasting experience to the UK population. Interview participant 1 described the importance for crews to stay aligned and aware of one another's work, despite being in different groups, to "[avoid] re-inventing the wheel", for example "someone will develop some capability in *Sport*, but when *iPlayer* comes around to implement it, it already exists, so [not keeping crews aligned is] a very costly mechanism".

3.4.2 Agile Development Approach. We understand that this department had used a mixture of different agile methods, but had transitioned from Scrum to a 'hybrid Kanban/Scrum' or 'Scrumban' approach (as described by interview participants) three years prior to our study. Interview participant 12 described the rationale for the change: "[we started] using Scrum… but what we found happened was that because we were working on a product that a number of other teams required or needed to feed into, it was very difficult to protect two weeks' worth of time, so we moved towards a more Kanban based model, as people often call it". However, Scrum ceremonies such as retrospectives and stand up meetings (or 'stand ups') were still used. Interview participant 12 described the benefit of the transition:

"We've transitioned from Scrum to more Kanban – I personally would find it very difficult to switch back to a Scrum based structure, I think the freedom you get from being in a continuous delivery flow means that it sort of just frees you up to focus on the problem and not have to worry about schedules or these kind of false promises you're making in a Scrum environment, where you're like 'oh I promise to do this by two weeks' time' rather than saying 'right, what's the highest priority thing?' So, from a developer's perspective I think it's very valuable to have that freedom to be able to say 'right, tell me the problem, state the goals, state the problem space and tell me my constraints and we'll go off and sort the problem, we'll go off and work out the best solution to that thing'"

Crews were encouraged to continually optimize and tailor their own processes to suit their needs. For example, they could choose to use digital and/or physical boards to visualize their work in progress. Participants noted the importance of flexibility and autonomy across the platform:

"Through the very nature of continuous improvement, we have layered on top of that various levels of Kanban and lean principles … and then the teams optimize in their own and unique way. I think that's encouraged … we aren't looking for standardization of everything, although some things may be standardized … what's right for today isn't right for tomorrow… the tools and approaches just aren't fit for tomorrow" – Interview participant 1

"If we don't give people autonomy, we might as well employ typists not developers" – Interview participant 12

"I think we are quite free … but we do need to align … Every crew needs a consistent front" – Observation participant 34

Given these descriptions, it was not expected that there would be one interpretation of how agile development should be practiced across the platform, although there were standardized practices in place. Individual crews even had their own manifestos written on paper and pinned onto their boards. One example observed was: 'we vow to: not break stuff, delete code that is not needed, visualise what we are doing, share knowledge so everyone can do everything'. This is also in accord with Schmidt's [61] analysis of Kanban systems at work and the need for a nuanced understanding of the ways in which formal organizational constructs such as plans, procedures, models are implemented in 'real time, real world' working practice.

3.5 Limitations

There are two particular and important limitations that arise from our study. Firstly, our case study was carried out at one organization that had been using 'a collection of agile methods' for around

4 years and this may not be representative of the challenges faced by more mature agile organizations. Secondly, the fact that the organization employed a range of different agile practices, which they described as their 'interpretation' of 'hybrid Scrum/Kanban' must be acknowledged before making generalizations to other large-scale agile practicing organizations.

4 FINDINGS

After carefully analyzing transcripts and observation data, the most prominent emerging theme was the tensions between standardization and flexibility in the large-scale agile environment. This insight was particularly interesting as TVaMP strived to foster a culture of flexibility within crews. However, there was a degree of standardization across the platform, which referred to processes that were enforced as a standard protocol to share knowledge and best practice. Tensions occurred when the desire to standardize certain processes prevented the ability to be flexible, whether this occurs within crews, from crew to crew or externally. Our findings report on a number of different processes and activities that agile crews engaged, with a focus on flexibility and standardization.

4.1 Work in Progress Boards

Physical and digital boards were a standardized mechanism used across the department to visualise work in progress (Fig. 2). These are traditional Kanban tools used to optimize work flow and provide visibility about the status of tasks to the crew and wider platform. Boards are typically broken down into different 'lanes', which reflect the development stage of a certain task. A typical lane used by all crews is the 'doing' lane, which contained all the current tasks in progress (the work that the crew are currently doing). Tasks are usually referred to as 'tickets' and physical boards depict these tickets as post-it notes. Tickets are placed inside lanes to indicate their status, and move across each lane as they progress towards completion. Completed tasks usually move to a 'done' lane (to indicate that the task has been completed). Some crews used more than one board and some preferred a combination of physical and digital boards. Crews could view each other's physical boards by walking around the platform and looking at them. Digital boards were projected on electronic screens across the platform and could be viewed in the same way, or accessed online.

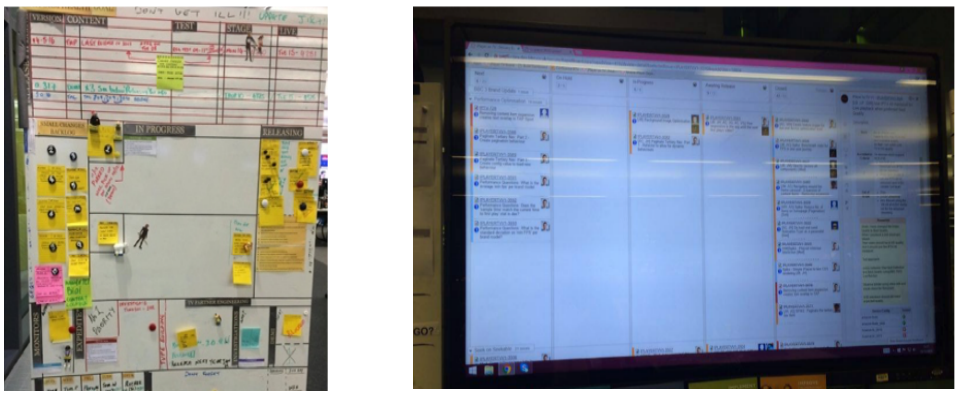

Fig. 2. A physical Kanban board (left) and a digital Kanban board (right).

4.1.1 Board Customization. Across the platform, each Kanban board displayed the same task information and used the same terminology to retain a universal understanding of progress, however crews could structure their boards in different ways. For example, all boards had a 'doing' and 'done' lane to represent development tasks which were in progress and those that had been completed. However, boards could have additional lanes such as 'releasing', which indicates that a

completed task (previously placed in the 'done' lane) is currently being released, or 'test' which indicates a ticket is currently in the testing phase of development. On physical boards, crews often conveyed information through handwritten post-it notes, magnets, personalized avatars (representing each crew member) and printed notes. Physical boards were divided in creative ways, providing the opportunity for crew members to place personal messages about their overall crew goals and reminders onto the boards for motivation. One example observed: "DON'T GET ILL!"

Digital boards were displayed on monitors near the desks of the crews. They were hosted on *Jira* or *Trello,* which are software project tracking and management tools that can be accessed online. Crews could choose which tool to use. The tools allowed users to define lanes and create tickets, similar to the physical boards. Digital boards typically had limited scope for rich visual customization compared to physical boards, as they were confined to the customization options offered by the software. Interview Participant 7 stated "I think they [physical boards] are far more flexible".

Physical boards varied in appearance, demonstrating how crews took full ownership of them. It was common practice for crew members across the platform to review one another's boards to understand their task progress, promoting visibility across the platform. Interview Participant 1 summarized the use and ownership of the boards across the platform: "We are all about radiating visibility about work and some people choose to do that on physical boards, some people choose to do that on electronic boards ... you can decide your own way of working, your own lane structure, whether you want just a simple 'to do, doing, done' [lane], whether you want more lanes than that."

4.1.2 Impacts of Board Choice on Work. Crews which chose to operate a combination of physical and digital boards faced significant overhead when attempting to sync the information across both boards to keep them consistent. Crews would have to replicate tickets from the physical board onto the digital board, as physical post-it notes that tracked information would eventually be thrown away, becoming irretrievable. Keeping a digital record of tickets was believed to provide stronger visibility and traceability across the platform, as crews could easily opt to access and search digital boards online rather than walking over to another crew's physical board. Participants discussed how a crew's board choice impacted their work and crew culture:

"There's a lot of conversations which happen around the [physical] board which I don't think would happen if everyone just used *Jira* and sat on their laptops" – Interview participant 5

"With physical boards, you cannot measure the history of what you did last week as tickets are pulled off [the board], which makes it hard to review your progress." – Observation participant 4

"It's impossible to keep digital and physical boards synced, when it's time to align them, the tickets go in the wrong orders and do not reflect the physical board." – Observation participant 10

Syncing the task information across different boards was especially difficult. Tickets from physical boards that were entered onto digital boards appeared in the order they were input onto the software, which did not always reflect the ticket's true status. For example, *Ticket-1* may appear after *Ticket-2* on a digital board, reflecting the time it was input onto the software rather than the actual ordering of the tasks. This caused confusion about progress between crews, making it difficult for crews to be coordinated and properly informed about one another's work. It was also difficult to capture certain information onto the digital boards from physical boards, as physical boards allowed for personalized notes and abstract representations of information (such as through avatars and magnets) which cannot be replicated in the same way onto *Jira* and *Trello*.

Some participants discussed feeling challenged by technical leads and PMs regarding the tasks which were on their boards. Observation participant 6 explained how this was dispiriting for crews, who need space and autonomy to fully engage with processes. This was later highlighted by technical leads during a meeting:

"It shows lack of trust in the group if they aren't putting stuff on the board because maybe they can't justify [the work] to management or don't feel like they can..." – Observation participant 7

"I seldom see boards that really reflect what is being played by crews" – Observation participant 8

In some instances crews were using boards which they did not find effective, but the board choice was made by the PM of their crew, instead of being a joint decision among the crew. A developer (observation participant 5) of a crew that used a digital board noted that they preferred physical boards, as they felt it was clearer to see task progress. They mentioned finding it more satisfying to move physical tickets across lanes rather than digital ones. Observation participant 5 discussed the drawbacks of using *Jira*: "*Jira* is a nightmare ... it's useful for statistics, but conversations happen over *Jira* when two people are sitting next to each other, just so it's all documented and up to date".

While observing a different crew who were discussing the status of a ticket, the PM requested that the developer documented their verbal conversation onto *Jira*. This was so their conversation regarding the status of a ticket was visible to the rest of the crew and platform. Crews that used a combination of physical and digital boards also had to document conversations onto their digital boards for the same reason, which was time consuming for the crew members.

The types of software tracking tools utilized for digital boards also had an impact on crews' work. A tester (observation participant 4) discussed rotating into a crew that operated their digital board on *Trello*. When asked why the crew had decided to use *Trello* instead of *Jira*, they mentioned that crew members had felt *Trello* was more attractive in terms of customization of the interface. However, it was also mentioned that *Trello* had caused some problems, such as having no concept of unique identifiers to refer to individual tickets. This caused particular problems for testers when raising bugs, and when crews did want to raise bugs and were using *Trello*, tickets had to be manually duplicated onto a separate *Jira* board so that unique identifiers were possible.

What these varied empirical findings point to are some of the subtleties and nuances involved in work organization systems. The important point of these findings, much as Schmidt [61] notes for Kanban systems, is that this is not a simple critique of any particular system but a critique in the Kantian sense: "an attempt to determine the proper domain of this approach, so as to unburden it of some popular misunderstandings and unwarranted generalizations and suggest some nuanced conceptualizations for further research".

4.2 Impact of Meetings

There were many different types of meetings utilized within crews and additionally to manage the large-scale environment at TVaMP. These included stand ups (a short, traditional Scrum style meeting), retrospective meetings (focused on improving and crews' work ethic and spirit), technology forums (to discuss convention when using technical platforms such as *Github*), three amigos (a meeting with the crew's PM, BA and developers to discuss starting a task), coalition meeting (opportunity to manage and review which crews are entering which code base) and generic work meetings. Of relevance here is Boden's [44] work in 'The Business of Talk' documenting the 'skill' routinely deployed in meetings that represents an accomplishment 'turn by turn and topic by topic'; whereby 'people... talk their way to solutions, talk themselves into working agreements... talk their organizational agendas ... create and recreate fine distinctions that actually make the organization come alive'.

4.2.1 The Effectiveness of Stand Ups. Stand ups were standardized across the platform, inherited from the platform's prior Scrum methodology. They happened daily at around 10am, lasting around 10 minutes. Stand ups had around 6-10 attendees, usually every crew member and occasionally additional attendees from management or other crews. Stand ups were designed for all crew members to discuss task status and highlight any potential challenges. Depending on the crew, stand ups were either chaired by rotating crew members or the PM. The chair would run through each task on the board and decide which was going to be completed next. Stand ups also served as a platform to have face-to-face discussion and seek clarity regarding comments left on digital boards.

Occasionally, stand ups were attended by members from different crews. A PM (observation participant 11) explained that attending other crews' stand ups was encouraged, as it promoted visibility across the platform and understanding of task status in short block of time. However, developers expressed feeling that their stand ups became complex and lacked value as a result of other crews and management attending them:

"Stand ups are management interference" – Observation participant 13

"Stand ups should be short and concise, there should be less emphasis on ceremonies and more emphasis on doing work … be surgical about stand ups, they become useless to the crew as too many people come" – Observation participant 3

"Stand ups are not agile. If you have a specific time to communicate in a slot that is prescribed, it shows you're not communicating enough during the day" – Observation participant 15

A 6 minute stand up was observed with 19 attendees. A developer (observation participant 17) from this crew stated that the size was due to merging 3 crews' stand ups to promote visibility between collaborating the crews. There was a tally chart on the physical board where members could vote on their ideal stand up size. Out of 13 votes, 10 voted to split the stand up into 2 groups, 1 voted to split into 4 groups and 2 voted to remain the same size. Crew members admitted that communicating became much harder when stand ups were large and were subsequently too short to be productive.

Observation participant 16 reflected on an old process where a group of 10 BAs and PMs would observe each stand up, every day. Crews would have their stand ups when the BAs and PMs were free to observe. This was a practice introduced as an experiment to promote visibility, but made crews feel that they had to "perform" (observation participant 16) stand ups to management rather than conducting them for their intended value. Crews ended up having their "real stand-ups" (observation participant 16) at the start of the day and performing a second stand up for management later that same day. This process was later dismissed as it was seen as time-consuming for both crews and management, and also made stand ups futile for crews.

4.2.2 Impact of Meetings on Productivity. A meeting was observed that was called to discuss the usefulness and quantity of meetings across the platform. There were 9 people in attendance. The meeting was motivated by crew members feeling distracted and unproductive due to the number of meetings that they had to attend. Developers and testers expressed their concerns:

"I've been to three meetings this week, with the same purpose, and the same people... No meetings would be like music to my ears" – Observation participant 18

"On Thursday I had no meetings, my pair and I got so much done" – Observation participant 20

"I've had so many distractions this week, I want to put earphones in, but then I can't pair program … [My other crew members] are constantly in meetings... I feel I have little time to do work" – Observation participant 19

It was observed that testers and developers felt distracted by the number of meetings and saw them as interrupting work flow. Multiple meetings were in place to keep the platform aligned and informed about the work of individual crews. Although each crew was independent, collectively they were working towards shared goals and broader projects which must be aligned. A trial period with fewer meetings for developers and testers was suggested at the meeting as a potential solution to this issue. There was debate around the necessity of including management and BAs into this trial, as "by nature they attend many meetings" (observation participant 18). A PM (observation participant 22) then stated, "a few less meetings won't mean much" to which a developer (observation participant 3) responded that less meetings would "really help developers out".

A developer (interview participant 25) later stated that the usefulness of meetings for the promotion of visibility and alignment in the large-scale environment depended on how crew members' viewed their job. They stated: "If you define my work as writing code then it's really a

disruption to that, but if you define it as making valuable products and writing code is a byproduct of that, then yeah, talking is good."

4.2.3 Retrospective Meetings. Retrospectives were individual crew meetings that occurred approximately once a month and lasted an hour. These meetings were also inherited from the platform's prior Scrum approach. They were open only to the individual crew and are conducted in "quite a lively way … involving just a general congratulations" (interview participant 25). The aim was to openly and honestly reflect on work and potential improvements, followed by ensuring their work contributed to the shared goals of the wider platform. The meeting usually incorporated several action points to be completed by the next retrospective. All crew members were encouraged to use this meeting to share their opinions on processes and crew culture. Crews would keep annotated post-it notes from the retrospective to review if targets have been met (see Fig. 3).

A PM reflected on a discussion from a retrospective. They noted that demands from external departments and other crews meant that their crew had to put some of their own work on hold:

"As soon as you have more than one team then the challenge of autonomy versus alignment is significant, because when you've got a single team doing a single thing that doesn't depend on anything else, it doesn't have to be aligned with anybody so you just kind of stand back and watch the magic happen, if you are 7 trying to co-ordinate as we are, across 5 crews in my space, but also across multiple crews, several on this floor, some in London, some in Cardiff, Glasgow, it becomes a real balance between autonomy and alignment." – Interview participant 14

This was echoed by a PM (observation participant 32) who revealed that he faced dependencies from another department who practiced Scrum methodologies: "60% of my work depends on getting stuff from [another department]". For example if TVaMP (who operate using a hybrid approach) requested a piece of work from 'department X' (who operated using Scrum), this request is placed at the end of department X's Scrum Product Backlog (a list of tasks which need to be done) and is not placed according to priority level (as is typical with Kanban approaches). This meant that TVaMP had to wait for other departments to finish their prior tasks before their request was considered and fulfilled. The lack of interfacing techniques between collaborating departments and crews created dependency and productivity challenges.

Fig. 3. Flipchart paper containing annotated post-it notes from previous retrospective meetings.

4.2.4 Demonstrations. A monthly demonstration or 'demo' was observed where each crew presented their work from that month to the rest of the platform. They were typically organized by management. The observed demo lasted 1 hour. A PM described the demo as a way to celebrate the work of each crew while keeping motivation and morale high. PM (observation participant 32) seemed excited and proud to have this activity observed. It was held on a Friday afternoon (the end of the working day) on the platform floor, along with beverages. Each crew had 3 minutes to present

their work and their plans for the month ahead. The process was laid-back and crew members seemed enthusiastic to present their work. There was an opportunity for questions and feedback after each presentation. While demos were motivating for crews, they also provided a single space to promote alignment and visibility across the platform. It was a chance for the whole platform to come together and understand what each crew was working on during the prior month, which ordinarily would have been difficult due to the number of crews on the platform. There were also proposed opportunities for collaborations with certain tasks, providing a single space to reach all crew members. Interview participant 1 discussed why they believed autonomy and collaboration is important for crews: "You get people really invested in what they've done if they feel they've had a chance to input into what the thing is, even if that input is limited".

These empirical findings illustrate how various meetings, combined with the use of different boards form part of the overall management – the 'phasing' [41] of the project. As a range of CSCW studies have illustrated [57, 58], such meetings involve reporting on progress, issues and concerns and are a means whereby team members can orient to the project as a whole and keep informed of both progress and problems. They chart the progress (or lack of) of the project; what tasks are delayed or reallocated, what further work needs to be done in terms of coordination and the allocation of responsibilities; and thereby provides some indication of how the team are doing and what remains to be done. So, meetings become a mechanism for ordering, sequencing, allocating managing and tracking progress.

5 DISCUSSION

The question posited by this study is how to balance this flexibility with standardization. We use this question to frame a discussion below around how agile development is achieved at the large-scale TVaMP, with a focus on the implications of both flexibility and the standardization.

5.1 Implications of Flexibility and Standardization

5.1.1. Customization and Overhead. Offering crews the choice to implement and customize their own Kanban boards promoted flexibility. Dikert et al. [27] emphasizes the importance of customization in agile implementation, with teams who customize and tailor their agile practices performing better than those that do not. Customization also promotes a new way of thinking [27], which can be central to development teams who are not used to possessing autonomy in software development environments [53]. We observed that although crews had the flexibility to choose their boards, there were notable drawbacks for crews who chose to implement digital boards or a combination of board types. Discussions had around physical boards were documented on digital boards to promote coordination and awareness across the platform. Similarly, hand-written tickets were replicated onto digital boards, which often led to inaccuracies and misinformation being projected due to the way these software tracking tools operated. This caused crews to spend time attempting to capture the richness of their face-to-face discussions to be able to accurately reflect their work, which effected time spent carrying out development duties. Bardram and Bossen [26] discuss how physical artifacts that support work are only truly comprehensible by those in the team in which the artifact belongs, with newcomers or onlookers finding it challenging to decipher inscriptions on the artifacts. This coincides with the reasoning behind the adoption of digital boards in our study – to attempt to standardize the way that task information was projected across the platform, as physical boards were 'too' customized and personal to the crew. Increased flexibility in the customization of physical boards subsequently made them less accessible or comprehensible to the wider platform. We saw how the platform attempted to mitigate this: through duplicating tickets and notes onto a more accessible and 'consistent' software tracking tool. In turn, this replaced the prior problem of inaccessibility with the problem of increased time in replicating information – often inaccurately or in the wrong order – for the purpose of visibility.

It was important for crews to continually keep the remainder of the platform informed about their work – without doing so would restrict knowledge sharing which was required for dependency management. However, crews were burdened with the laborious task of 'blending' [26] the digital

and the physical artifacts to maintain coordination and visibility in the large-scale environment. In this example, the flexibility to customize boards paired with the need for boards to work at scale drew crew members into a challenging cycle of having to standardize their own customizations for the rest of the platform. Crews were supporting the difficult maintenance of the two boards types, rather than the boards supporting the cooperative work of the crews. The difficulties faced by crews to capture the detail and customization offered by physical boards within digital boards resonates with work by Bardram and Bossen [26], who note that annotations on physical artefacts cannot be modelled by computer systems in the same way. One way to alleviate these challenges with aligning physical and digital boards in this context is to not treat the boards as if they *should* be exact copies of one another. Digital boards, used alongside physical boards solely for visibility, should not need to capture the same level of richness and abstraction offered by physical boards – their differences should be appreciated. Instead, differentiating the level of detail and required knowledge to be shared *within* crews and *between* crews can help to realise what is necessary for digital boards to contain and what can remain pertinent to the immediate crew on a physical board. This is one way to help preserve flexibility, enabling crews to still take ownership and customize their physical boards, while controlling and refining what is necessary for external presentation. Designers of software tracking tools should incorporate the desire for customization with the requirement for orderliness (allowing a more flexible way to input and order tickets, while attributing them to unique identifiers) in their tool designs.

5.1.2. Boards Replacing Face-to-Face Communication. Crews did not always use boards which they found effective. Maintenance, synchronization, documentation and duplication were examples of overhead caused through digital board use (and a combination of using both digital and physical boards). A notable issue with digital boards was the way they limited the amount of face-to-face communication within crews. Prioritizing individuals and interactions over processes and tools is core to agile development [1]. Using a combination of digital and physical boards forced crews to spend time documenting verbal conversations onto digital boards to remain visible to the rest of the crew and platform. Dikert et al. [27] noted the difficulty of striking a balance between self-organizing agile teams focusing on their own goals and those of the broader organization. Lack of respect for the wider organizational context can result in coordination challenges [27]. Here we see an imbalance where some crews were choosing tools that would best suit the wider environment, putting this before the benefit of the individual crew.

Of importance here is Heath and Luff's [38] finding on how information flow between colleagues should *complement* information presented by technology, as relying on information presented solely by the technology alone can leave room for misinterpretation. In our study, there was an expectation that the larger environment should periodically check crews' boards to see their progress, with digital boards being viewed as more accessible. There is, however, a need to welcome tools in the management of interactions in large-scale agile environments as noted by Murphy et al. [49]. Furthering this, Layman et al. [47] note that tools are "only as useful as the users make [them]" highlighting that it is essential that information displayed in project management tools are accurate and up to date for the viewing of all stakeholders. Our findings showed crews' desire to maintain near accurate documentation on digital boards had a negative effect on inter-team communication. This demonstrates the burden placed on crews associated with ensuring that these tools are constantly updated and reflect a degree of accuracy down to individual conversations.

5.1.3. Interfacing Different Agile Methods. Large-scale agile environments that encourage flexibility require well though-out mechanisms to effectively interface different agile methods. The difficulty of interfacing with non-agile teams has been noted by Begel and Nagappan [50], with scheduling mutual deliverables being a major challenge. Here we discuss the lack of consistent interfacing techniques between *agile* teams and departments, creating dependency and productivity issues for inter-departmental collaborations when different agile flavors are mixed and matched. Similar to this are the observations made by Hoda et al. [53] relating to task dependencies, where

cycles of work (known as 'sprints' in Scrum methodologies) result in cancellation due to too long waits for dependencies from third parties.

Some organizations create standards to overcome the problems of interfacing agile interpretations [27]. This is necessary to mitigate against interpretations being too different, which can cause friction when team members move to new teams [27], as experienced by observation participant 4 who was rotated into a crew that used *Trello* instead of *Jira*, causing difficulties for their testing duties. However, cooperative protocols within the work place, for example Kanban boards, inevitably encounter situations where the system becomes 'beyond its bounds' [37] and requires flexibility to adapt to a given situation, presenting potential difficulties for such organizational interfacing standards. Cooperative tools that are adapted to suit new situations may no longer conform to the defined interfacing standards outlined by the organization. There is a clear need to allow flexibility to adapt cooperative tools, whether for new contexts or a crew's desire to experiment and customize, but this must be balanced with an element of interfacing standardization to manage dependencies and scale. This interoperability issue is a relatively underexplored area of research which does not seem to be fully resolved in practice.

In instances where agile methods are interfaced between crews, such as with Kanban boards, it is important to note that the work visualized on these tools (both physical and digital) may not accurately reflect what a crew is working on. We observed how some crews felt they could not publish their progress on their boards, as they felt somewhat challenged by BAs and management about what they were working on. Therefore, boards did not always reflect their current work, creating a mismatch between what was presented on the board and what was being done in practice. In this way, some crews were unable to effectively engage with their chosen tools, limiting true crew autonomy and contributing to difficulties keeping the platform coordinated. Begel and Nagappan [50] noted similar feelings of discomfort in reporting progress from developers in the context of meetings, where they felt pressured to label a task as 'done'. Hoda et al. [53] observed the importance of trust between PMs in agile teams in order to successfully collaborate with management activities. Trust is explicitly noted as a core principle of the Agile Manifesto [1]. Here we see the negative effects that this discomfort and 'lack of trust' (as phrased by observant participant 7) places on the development teams and wider platform: inability to fully engage with their tools and projection of misinformation around task status.

5.1.4. Repurposing Practices to Manage Scale. Stand ups were enforced across the platform, carried over from the platform's prior pure Scrum approach. Dikert [27] observed that tensions that can arise when old and new methods are used side by side. Short, concise stand ups were favored. Large stand ups or having them observed in the name of coordination and awareness made them ineffective – or even "useless" for developers. In this sense, standardizing and repurposing stand ups as an activity to manage scale placed burden onto individual crews. The very description of *performing* a stand up suggested that crews viewed them as a platform to demonstrate how they were meeting the priorities and needs of management, rather than having a true discussion of the needs and priorities of the crew. Observing different stand ups served as a way to navigate through the progress of the large-scale environment. However, standardizing stand ups and the protocol of observing them was not beneficial to standardization *or* flexibility – or even considered agile by some participants. In much the same way as crews feeling uncomfortable publishing their progress on their boards, similar levels of discomfort could be expected during observed stand ups. This can lead crews to feeling that they cannot have the freedom to discuss their shared priorities and task. In Layman et al.'s [48] empirical study of an XP environment (a type of agile software development methodology), developers reported finding stand ups beneficial due to their contribution to project understanding and visibility. Similarly in Begel and Nagappan's study [50], daily stand ups were viewed as instrumental to communication and coordination by testers and developers, providing a platform for them to be brought together. However, interestingly in [48], the authors note that pair programming activities were seen as valuable only when they were not enforced, in which they were then perceived as an inefficient use of time, demonstrating how the enforcement of practices can contribute to loss of value. This is mirrored through the enforcements of stand ups and their

conduct in our study. Crews felt a similar frustration noted by Dikert et al. [27], when agile is simply perceived as the use of specific tools, with little understanding of the core concepts that bring value to these tools. It was clear that the value behind stand ups became lost to individual crews. Enforcement of practices and tools must be considered carefully, as this can lead to their over-use, contributing to these practices being viewed as time-worn and losing effectiveness.

5.1.5. Meetings and Standardization. The effect that different definitions of 'work' had on crews' attitudes towards meetings was notable. For instance, defining 'work' as "making valuable products" made meetings more acceptable, whereas defining work as "writing code" had the opposite effect. This demonstrates how language framing can deeply impact processes and their outputs [25]. The use of the expression "autonomy and alignment" (interview participant 14) as an alternative to 'flexibility and standardization' highlighted the juxtaposition of personal agency versus institutional control. In other words, autonomy is conceptually linked to individuals or groups, whereas flexibility is more related to processes and outputs.

Multiple meetings that are in place to maintain alignment and visibility across the platform were viewed as repetitive and impeding productivity by development teams. Previous empirical software studies have noted the perceived problem of distraction with numerous meetings in agile environments [27, 30, 50]. Particularly Begel and Nagappan [50], who noted that developers perceived Scrum meetings as distracting and inefficient. In our study, meetings were viewed as useful to those in BA and PM roles, linking with Murphy et al.'s [49] finding that PMs believed 'awareness of other people's work' is the highest benefit of agile development. Although the framing of language influenced developers' perceptions of meetings, the consensus among participants was that the *quantity* of meetings was the larger problem. Standardized meetings were beneficial for scale management, as they periodically brought different crews and roles together to discuss different topics, but this was at the expense of crew flexibility. The numerous meetings enforced across the platform negatively affected the morale and productivity of those in development roles. Hoda et al. [51] observed two instances of how agile teams dedicated a member of their development team or BAs to act as a proxy to communicate with customers for requirements and feedback, working particularly well for those with heightened communication skills. One imagines how a crew member in this instance (possibly a PMs or BAs, who are most enthusiastic about meetings) could serve as a proxy to communicate with the wider platform through the frequent meetings aimed at managing scale. This should not mean that other crew members are exempt from meetings, as they do show to be beneficial to development team members, and face-to-face communication is important for understanding context and circumstances [48, 50, 34]. Furthering this, excluding developers and testers from meetings could affect their perception of their own responsibility and autonomy within projects. However, a proxy can help to manage the high frequency of meetings required for scale. It was clear from our findings that developers felt that less meetings would "really help them out" whereas a decrease in meetings was thought to have a minimal effect on PMs. Utilizing a dedicated proxy could help to promote face-to-face discussions in preparing the proxy for these meetings, alleviating tensions around lack of face-to-face communication within crews. Crew members may feel more comfortable discussing progress to their within crew proxies, addressing discomfort around this progress [50].

There were two examples of standardized meetings that were beneficial for flexibility in our study. These were demonstrations and retrospectives. Retrospectives, a standardized meeting *within* crews, successfully promoted flexibility through providing a dedicated time for crews to gather independently and reflect honestly on their processes and culture. They also served as an opportunity for crews to discuss how their work contributes to the overall shared goals of the broader organization. These were closed meetings were not observed, which may have contributed to their success. Realigning crew values and processes, followed by broader goals and values of the platform through standardized retrospectives had a positive impact on standardization *and* flexibility. Likewise, demonstrations proved to be an excellent opportunity to promote visibility and coordination whilst boosting motivation across the large-scale platform in a flexible, relaxed

way. They were highly positive for standardization, providing a single platform for all crews to present their work and gain feedback.

6 CONCLUSION

Through 17 interviews and approximately 54 hours of direct observation, our study has shown the tensions that manifest when balancing standardization and flexibility in the context of large-scale agile development. It is known that software development at scale comes with a number problems and challenges, and our work provides a detailed account of how and why these problems arise from those directly embedded in the agile process. We highlight five key findings associated with flexibility and standardization in this context, relating to: customization and overhead; Kanban boards and communication; interfacing agile methods; repurposing agile methods to manage scale and meetings and standardization. We argue large-scale organizations must strive to encourage processes, tools and value that do not overlook the importance of flexibility with an eye to increasing standardization. We demonstrate the specific difficulties that can arise as a result of an imbalance of flexibility and standardization. Since work activities are inherently flexible, subtle and contextual; technologies and systems to support work, CSCW technologies and systems, need to possess similar qualities. One of the main achievements of CSCW has been to empirically document the complexities of collaborative socio-technical work and thereby delineate what some regard as the classic CSCW problems, 'what to automate and what to leave to human skill and ingenuity' [29]. The contrast and tension of 'standardization' versus 'flexibility' constitutes an extension of, or complement to, this classic dilemma.

To some extent the empirical work reported here is similar to that found in research on development projects more generally – that is, it is a simple addition to a growing empirical research corpus. It reinforces the idea that software development work is complicated and intricate, and subject to negotiation and renegotiation. Development work is inevitably a 'satisficing' activity, it requires the discovery and achievement of workable and acceptable compromises. However, turning this into a 'theory' is beyond the remit of this paper, but we certainly believe it can contribute to some of the conceptual developments noted by Hoda et al. [54, 53, 51] in particular the concept of 'balancing'. As they note, Glaser's [59] notion of 'balancing' – "handling many variables at once in order to start an action, keep an action going or achieve a resolution" - best fitted their own findings concerning the practices of self-organizing agile teams. We believe the concept of 'balancing' might also fit with some of our findings concerning flexibility and standardization; and whilst far from being a 'theory' in any rigorous sense, may still contribute to what Halverson [56] considers some of the attributes of theory, in particular in providing a useful way of talking about and describing some phenomena, and thereby provoking some ideas or 'implications for design'. As Wittgenstein [60] notes, "The difficulty – I might say – is not that of finding a solution but rather as recognizing as the solution something that looks as if it were only a preliminary to it… This is connected, I believe, with our wrongly expecting an explanation, whereas the solution of the difficulty is a description, if we give it the right place in our consideration".

ACKNOWLEDGEMENTS

We would like to thank Duncan Fortescue and all our participants at the BBC for their involvement in our study. Also, we thank Mark Rouncefield for his valuable input into our work. This work has received support from the 'Values in Computing' project funded by the Engineering and Physical Sciences Research Council (EPRSC) UK, Grant number: EP/R009600/1).

REFERENCES

[1] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Rob Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas. 2013. The Manifesto for Agile Software Development. Retrieved March 26, 2018 from http://agilemanifesto.org

[2] Scott W. Ambler. 2007. Agile Software Development at Scale. In *Balancing Agility and Formalism in Software Engineering,* Meyer B., Nawrocki J.R., Walter B. (Eds.) Lecture Notes in Computer Science, Vol 5082. Springer, Berlin, Heidelberg. 1-12. DOI: 10.1007/978-3-540-85279-7_1

[3] Taghi Javdani Gandomani and Mina Ziaei Nafchi. 2015. An Empirically-Developed Framework for Agile Transition and Adoption: A Grounded Theory Approach. *Journal of Systems and Software*, Benoit Baudry, Antonia Bertolino, Daniela Damian, Kaushik Dutta, Helen D. Karatza, Patricia Lago (Eds.) Vol 107. Elsevier Science Inc, New York. 204-219. DOI: 10.1016/j.jss.2015.06.006

[4] Shventha Soundararajan, James D Arthur and Osman Balci. 2012. A Methodology for Assessing Agile Software Development Methods. In *Proceedings of Agile 2012 Conference* (AGILE '12). 51-54.

[5] Ipek Ozkaya, Michael J. Gagliardi and Robert Nord. 2013. Architecting for Large-scale Agile Software Development: A Risk-Driven Approach. In *Crosstalk Magazine*, Brandon Ellis and Colin Kelly (Eds.) Vol 26. Software Engineering Institute. 17-22.

[6] Jutta Eckstein. 2004. *Agile Software Development in the Large: Diving into the Deep*. Dorset House Publ. Co., Inc., New York, NY, USA.

[7] Torgeir Dingsøyr, Tor Erland Fægri and Juha Itkonen. 2014. What Is Large in Large-Scale? *A Taxonomy of Scale for Agile Software Development*. In Product-Focused Software Process Improvement, Jedlitschka A., Kuvaja P., Kuhrmann M., Männistö T., Münch J., Raatikainen M. (Eds.) Lecture Notes in Computer Science, Vol 8892. Springer, Cham. 273-276. DOI: 10.1007/978-3-319-13835-0_20

[8] Nirnaya Tripathi, Pilar Rodriguez, Muhammad Ovais Ahmad and Markku Oivo. 2015. Scaling Kanban for Software Development in a Multisite Organization: Challenges and Potential Solutions. In *Agile Processes in Software Engineering and Extreme Programming* (XP '15), Lassenius C., Dingsøyr T., Paasivaara M. (Eds.) Lecture Notes in Business Information Processing, Vol 212. Springer, Cham. 178-190. DOI: 10.1007/978-3-319-18612-2_15

[9] Peggy Gregory, Leonor Barroca, Katie Taylor, Dina Salah and Helen Sharp. 2015. Agile Challenges in Practice: A Thematic Analysis. *In Agile Processes in Software Engineering and Extreme Programming* (XP '15), Lassenius C., Dingsøyr T., Paasivaara M. (Eds.) Lecture Notes in Business Information Processing, Vol 212. Springer, Cham. 64-80. DOI: 10.1007/978-3-319-18612-2_6

[10] Jo E. Hannay and Hans Christian Benestad. 2010. Perceived Productivity Threats in Large Agile Development Projects. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (ESEM '10). Article 15. DOI: 10.1145/1852786.1852806

[11] Sallyann Freudenberg and Helen Sharp. 2010. The Top 10 Burning Research Questions from Practitioners. In *IEEE Software*, Vol27. IEEE Computer Society Press Los Alamitos, CA. 8-9. DOI: 10.1109/MS.2010.129

[12] Lech Krzanik, Pilar Rodriguez, Jouni Simila, Pasi Kuvaja and Anna Rohunen. 2010. Exploring the Transient Nature of Agile Project Management Practices. In *2010 43rd International Conference on System Sciences* (HICSS '10). DOI: 10.1109/HICSS.2010.204

[13] Nilay Oza, Fabian Fagerholm and Jürgen Münch. 2013. How Does Kanban Impact Communication and Collaboration in Software Engineering Teams? In *Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering* (CHASE 2013). 125-128. DOI: 10.1109/CHASE.2013.6614747

[14] Dan Turk, Robert France and Bernhard Rumpe. 2002. Limitations of Agile Software Processes. In *Third International Conference on Extreme Programming and Flexible Processes in Software Engineering* (XP '02) 43-46. Springer, Verlag.

[15] Eman A. Altameem. 2015. Impact of Agile Methodology on Software Development. In *Computer and Information Science. Canadian Center of Science and Education*, Vol 8. 1-10. DOI: 10.5539/cis.v8n2p9

[16] Alan Koch. 2011. 12 Advantages of Agile Software Development. Retrieved March 26, 2018 from https://www.globalknowledge.com/ca-en/resources/resource-library/white-paper/12-advantages-of-agile-software-development/

[17] Elizabeth Whitworth. 2008. Experience Report: The Social Nature of Agile Teams. In *Proceedings of Agile 2008 Conference* (AGILE '08). 3-6. DOI: 10.1109/Agile.2008.53

[18] Christiane Gresse von Wangenheim, Jean Carlo R. Hauck, Alessandra Zoucas, Clenio F. Salviano, Fergal McCaffery and Forrest Shull. 2010. Creating Software Process Capability/Maturity Models. *IEEE Software,* Vol 27. 92-94. DOI: 10.1109/MS.2010.96

[19] Torgeir Dingsøyr, Tore Dybå and Pekka Abrahamsson. 2008. A Preliminary Roadmap for Empirical Research on Agile Software Development. In *Proceedings of Agile 2008 Conference* (AGILE '08). 3-6. DOI: 10.1109/Agile.2008.50

[20] Dag I. K. Sjøberg, Tore Dybå and Magne Jørgensen. 2007. The Future of Empirical Methods in Software Engineering Research. In *Future of Software Engineering* (FOSE '07). 358-378 DOI: 10.1109/FOSE.2007.30

[21] Torgeir Dingsøyr and Nils Brede Moe. 2013. Research Challenges in Large-Scale Agile Software Development. ACM SIGSOFT Software Engineering Notes. Vol 38, no 5. 38-39. DOI: 10.1145/2507288.2507322

[22] Virginia Braun and Victoria Clarke. 2006. Using Thematic Analysis in Psychology. In *Qualitative Research in Psychology*. Vol 3. 77-101. DOI: 10.1191/1478088706qp063oa

[23] Saul McLeod. 2015. Observation Methods. Retrieved from May 25, 2017 from www.simplypsychology.org/observation.html

[24] Craig Larman and Bas Vodde. 2010. Practices for Scaling Lean & Agile Development: Large, Multisite and Offshore Product Development with Large-Scale Scrum. Addison-Wesley Professional.

[25] George Lakoff. 2010. Why it Matters How We Frame the Environment. *Environmental Communication*. Vol 4, no 1. 70–81. DOI: 10.1080/17524030903529749

[26] Jakob E. Bardram and Claus Bossen. 2005. A Web of Coordinative Artifacts: Collaborative Work at a Hospital Ward. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work* (GROUP '05). 168-176. DOI: 10.1145/1099203.1099235

[27] Kim Dikert, Maria Paasivaara and Casper Lassenius. 2016. Challenges and Success Factors for Large-Scale Agile Transformations. A Systematic Literature Review. *Journal of Systems and Software,* Hans van Vliet*,* Benoit Baundry, Antonia Bertolino, Daniela Damian, Juan C. Dueñas López, Kaushik Dutta, Helen D. Karatza, Patricia Lago (Eds.) Vol 119. Elsevier Inc. 87-108. DOI: 10.1016/j.jss.2016.06.013

[28] Paul Hodgetts. 2004. Refactoring the Development Process: Experiences with the Incremental Adoption of Agile Practices. In *Agile Development Conference.* 106-113. DOI: 10.1109/ADEVC.2004.17

[29] Dan Shapiro. 1994. The Limits of Ethnography: Combining Social Sciences for CSCW. In *Proceedings of the 1994 ACM conference on Computer Supported Cooperative Work* (CSCW '94). 417-428. DOI: 10.1145/192844.193064

[30] Andrew Begel and Nachiappan Nagappan. 2007. Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. In *Empirical Software Engineering and Measurement* (ESEM '07). 255-264. DOI: 10.1109/ESEM.2007.12

[31] Paul Dourish and Victoria Bellotti. 1992. Awareness and Coordination in Shared Workspaces. In *CSCW '92 Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work* (CSCW '92). 107-114. DOI: 10.1145/143457.143468

[32] Paul Dourish and Sara Bly. 1992. Portholes: Supporting Awareness in a Distributed Work Group. In *CHI '92 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '92). 541-547. DOI: 10.1145/142750.142982

[33] Helen Sharp, Rosalba Giuffrida and Grigori Melnik. 2012 Information Flow within a Dispersed Agile Team: A Distributed Cognition Perspective. In Wohlin C. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2012. Lecture Notes in Business Information Processing, vol 111. Springer, Berlin, Heidelberg

[34] Elihu M Gerson. 2008. Reach, Bracket, and the Limits of Rationalized Coordination: Some Challenges for CSCW. In *Resources, Co-Evolution and Artifacts. Computer Supported Cooperative Work.* Springer-Verlag, London 2008. DOI: 10.1007/978-1-84628-901-9_8

[35] Kate Kuusinen, Peggy Gregory, Helen Sharp, Leonor Barroca, Katie Taylor and Laurence Wood. 2017. Knowledge Sharing in a Large-Agile Organisation: A Survey Study. In Baumeister H., Lichter H., Riebisch M. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2017. Lecture Notes in Business Information Processing. Vol 283. Springer, Cham.

[36] Knut H. Rolland, Vidar Mikkelsen and Alexander Næss. 2016. Tailoring Agile in the Large: Experience and Reflections from a Large-Scale Agile Software Development Project. In: Sharp H., Hall T. (eds) Agile Processes, in Software Engineering, and Extreme Programming. XP 2016. Lecture Notes in Business Information Processing. Vol 251. Springer, Cham.

[37] Kjeld Schmidt and Carla Simone. 1996. Coordination Mechanisms: Towards a Conceptual Foundation of CSCW Systems Design. In *Computer Supported Cooperative Work: The Journal of Collaborative Computing.* Vol 5. 155-200. Kluwer Academic Publishers.

[38] Christian Heath and Paul Luff. 1992. Collaboration and Control Crisis Management and Multimedia Technology in London Underground Line Control Rooms. In *Computer Supported Cooperative Work* (CSCW '92). Vol 1, no 1-2. 69-94. DOI: 10.1007/BF00752451

[39] Yvonne Dittrich. 2002. Doing Empirical Research on Software Development: Finding a Path between Understanding, Intervention and Method Development. In *Social Thinking-Software Practice*. 243-262.

[40] Wes Sharrock and Kjeld Schmidt. 1996. Introduction: Studies of Cooperative Design. In *Computer Supported Cooperative Work: The Journal of Collaborative Computing*. Vol 5. 337-339.

[41] Graham Button and Wes Sharrock. 1994. Occasioned Practices in the Work of Software Engineers. In *Requirements Engineering*. 217-240. Academic Press Professional Inc.

[42] Rebecca E. Grinter. 1998. Recomposition: Putting it all Back Together Again. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative* Work (CSCW '98). Vol 5. 337-339.

[43] Rob Procter, Mark Rouncefield, Meik Poschen, Yuwei Lin and Alex Voss. 2011. Agile Project Management: A Case Study of a Virtual Research Environment Development Project. *In Computer Supported Cooperative Work* (CSCW '11). Vol 2, no 3. 197-225. DOI: 10.1007/s10606-011-9137-z

[44] Deirdre Boden. 1994. The Business of Talk: Organizations in Action. *Cambridge: Polity Press.*

[45] Lucy A Suchman. 1987. Plans and Situated Actions: The Problem of Human-Machine Communication. Cambridge University Press.

[46] John Hughes, Val King, Tom Rodden and Hans Anderson. 1994. Out of the Control Room: The Use of Ethnography in Systems Design. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work* (CSCW '94). 22-26. DOI: 10.1145/192844.193065

[47] Lucas Layman, Laurie Williams, Daniela Damian and Hynek Bures. 2006. Essential Communication Practices for Extreme Programming in a Global Software Development Team. Information & Software Technology 48(9): 781-794. DOI: 10.1016/j.infsof.2006.01.004

[48] Lucas Layman, Laurie A. Williams and Lynn Cunningham. 2004. Exploring Extreme Programming in Context: An Industrial Case Study. Agile Development Conference. 32-41. DOI: 10.1109/ADEVC.2004.15

[49] Brendan Murphy, Christian Bird, Thomas Zimmermann, Laurie Williams, Nachiappan Nagappan and Andrew Begel. 2013. Have Agile Techniques been the Silver Bullet for Software Development at Microsoft? *Proceedings of the Seventh International Symposium on Empirical Software Engineering and Measurement.* 75-84.

[50] Andrew Begel and Nachiappan Nagappan. 2007. Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. *First International Symposium on Empirical Software Engineering and Metrics.* 255-264.

[51] Rashida Hoda, Philippe Kruchten, James Noble, and Stuart Marshall. 2010. Agility in Context. Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Application. 74-88. DOI: 10.1145/1869459.1869467

[52] Philippe Kruchten. 2013. Contextualizing Agile Software Development. Journal of Software: Evolution and Process, 25(4). 351-361. DOI: doi.org/10.1002/smr.572

[53] Rashida Hoda and Latha K. Murugesan. 2016. Multi-level Agile Project Management Challenges: A Self-Organizing Team Perspective. Journal of Systems and Software. 117. 245-257. DOI: 10.1016/j.jss.2016.02.049

[54] Rashida Hoda, James Noble and Stuart Marshall. 2012. Developing a Grounded Theory to Explain the Practices of Self-Organizing Agile Teams. Empirical Software Engineering, 17(6) .609-639. DOI: doi.org/10.1007/s10664-011-9161-0

[55] Jeffry S. Babb, Rashina Hoda, and Jacob Nørbjerg. 2014. XP in a Small Software Development Business: Adapting to Local Constraints. In Scandinavian Conference on Information Systems. 14-29. DOI: doi.org/10.1007/978-3-319-09546-2_2

[56] Christine A. Halverson. 2002. Activity Theory and Distributed Cognition: Or What Does CSCW Need to DO With Theories? *Computer Supported Cooperative Work (CSCW)*, *11*(1-2). 243-267.

[57] Dave Martin, John Mariani, and Mark Rouncefield. 2004. Implementing an HIS project: Everyday Features and Practicalities of NHS Project Work. *Health Informatics Journal*, *10*(4). 303-313.

[58] John Bowers. The Work to Make the Network Work: Studying CSCW in Action. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* 287–98. ACM Press, 1994.

[59] Barney G. Glaser. 2005. The Grounded Theory Perspective III: Theoretical Coding. Sociology Press.

[60] Ludwig Wittgenstein. 1967. Zettel Oxford: Blackwell, 1967. Quoted in Rowe, M.W., 1991. Goethe and Wittgenstein. *Philosophy*, *66*(257). 283-303.

[61] Kjeld Schmidt. 1997. Of Maps and Scripts - the Status of Formal Constructs in Cooperative Work. *In Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work: the Integration Challenge* (GROUP '97). ACM, New York, NY, USA. 138-147. DOI: http://dx.doi.org/10.1145/266838.266887