FACULTY OF ENGINEERING AND TECHNOLOGY MASTER OF SOFTWARE ENGINEERING

Investigating the effect of monitoring productivity factors on global Agile teams' productivity

دراسة تأثير مراقبة عوامل الانتاجية على انتاجية فرق تطوير البرمجيات العالمية المتّبعة للنهج السريع في تطوير البرمجيات

Author:
Emad Ashour
(1155348)

Supervisor:
Dr. Yousef Hassouneh

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in Software Engineering at Birzeit University, Palestine

June 12, 2019

جامعة بيرزيت
BIRZEIT UNIVERSITY

Approved by the thesis committee:

Dr. Yousef Hassouneh, Birzeit University

Dr. Nariman Ammar, Birzeit University

Dr. Sobhi Ahmed, Birzeit University

Date approved:

Declaration of Authorship

I, Emad Ashour (1155348), declare that this thesis titled, “ Investigating the effect of monitoring productivity factors on global Agile teams’ productivity

دراسة تأثير مراقبة عوامل الانتاجية على انتاجية فرق تطوير البرمجيات العالمية المتّبعة للنهج السريع في تطوير البرمجيات

” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master degree at Birzeit University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Abstract

Productivity is a topic that discussed widely in literature, and has its specialty since no certain definitions were agreed upon. Productivity also has its specialty when talking about teams with unconventional settings, such as the teams working in distributed geographical locations, where the face-to-face communication is absent, which causes conflicts that might lead to impact the productivity of the team. Additionally, Agile methodologies were adopted by many companies around the world, benefiting the flexibility and the customer satisfaction. One of the popular Agile frameworks is Scrum, which originally based on the face-to-face communication among the team members to enhance the communication and the conflict resolution. However, the lack of the literature studies discussing and proposing ways to enhance the productivity in the Scrum teams, distributed among different geographical locations, the importance of this study is emerged.

This study proposes a mechanism that could enhance the productivity of Scrum distributed teams, and uses some productivity factors, collected from the literature, which were proven to have impact on the productivity of Software teams. It also proposes a feedback mechanism, to allow the managers of Software teams to monitor and to act based on the collected feedback from the team members. To investigate this, a prototype is developed based on a proposed feedback framework, and studied using exploratory research methodology: case study, that is suitable for such kind of researches. The study discovered an enhancement on the productivity of the participated teams, also found that the feedback taken on specific productivity factor, could enhance that factor across the team, provided the suitable action by the manager of the team.

ملخص

الإنتاجية هي موضوع نوقش على نطاق واسع في أدبيات هندسة البرمجيات ، وله خصوصيته لأنه لم يتم الاتفاق على تعاريف معينة لهذا المصطلح. تتميز الإنتاجية أيضًا بخصوصية عند الحديث عن فرق برمجيات ذات إعدادات غير تقليدية ، مثل الفرق العاملة في مواقع جغرافية موزعة ، حيث يتغيب التواصل المباشر، مما يؤدي إلى تعارضات قد تؤدي إلى التأثير على إنتاجية الفريق. بالإضافة إلى ذلك، تم اعتماد منهجيات التطوير السريعة من قبل العديد من الشركات في جميع أنحاء العالم، للإستفادة من المرونة و تحقيق رضاء العملاء. واحدة من أطر عمل منهجيات التطوير السريعة الشائعة هي صخرم ، والتي تستند أساسًا إلى التواصل المباشر بين أعضاء الفريق لتعزيز التواصل وحل التعارض. ومع ذلك، فقد ظهرت أهمية هذه الدراسة في عدم وجود دراسات في الأدبيات لمناقشة واقتراح طرق لتعزيز الإنتاجية في فرق التطوير اللي تتبع المنجيات السريعة لتطوير البرمجيات، موزعة بين مختلف المواقع الجغرافيةز تقترح هذه الدراسة آلية يمكن أن تعزز إنتاجية فرق صخرم ، وتستخدم بعض عوامل الإنتاجية ، التي تم جمعها من الأدبيات، والتي ثبت أن لها تأثير على إنتاجية فرق البرمجيات. ويقترح أيضًا آلية للتغذية الراجعة ، للسماح لمديري فرق البرمجيات بمراقبة التغذية الراجعة التي تم جمعها من أعضاء الفريق والتصرف بناءً عليه. لدراسة ذلك ، تم تطوير نموذج أولي بناءً على إطار التغذية الراجعة المقترح ، ودُرس باستخدام احدة منهجيات البحث الاستكشافي المعروف باسم دراسة الحالة ، وهي مناسبة لمثل هذا النوع من الأبحاث. اكتشفت الدراسة تحسنًا في إنتاجية الفرق المشاركة ، كما وجدت أن التغذية الراجعة التي تم جمعها لعامل إنتاج معين ، يمكن أن تعزز هذا العامل في جميع أنحاء الفريق ، شريطة اتخاذ الإجراء المناسب من قبل مدير الفريق.

Acknowledgements

To my family, my wife who supported me all the time to complete my master degree, my children, parents and siblings. I would like to especially thank my company and Mr. Tareq Maayah for his support, Dr. Yousef Hassouneh and all of the teaching crew of Master in Software Engineer program at Birzeit University for their great efforts and continuous support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Agile has been widely adopted as development process that achieved many success stories since it was appeared in early early 2000. Currently, Agile approaches are better software development process [11], since its flexibility that focuses on individuals and interactions, minimizes documentation work, involves customer in the development process and accepts requirements changes. Global Software Development (GSD) has been emerged as result of Globalization in business. The main reason for adopting GSD is to have access to lower cost skilled human resources [16]. However, GSD has many challenges that affect GSD process such as communication, coordination and trust [30, 43, 19], which affect the performance of GSD teams. Agile methodologies have been adopted by GSD due to its features, such as customer satisfaction, and short time to market[13]. Scrum is a framework applies the Agile principles and used in GSD teams, although using Scrum has showed success, the combination between GSD and Scrum is still not clear [18]. On the other hand, adopting Agile in GSD encounters some challenges that GSD normally encounters, such as geographic distance, Socio-culture distance, temporal distance, language differences and ineffective communication [12]. In order to have a successful adoption to Agile methodologies in GSD, many improvements should be done to overcome some of the challenges in GSD [18]. The challenges are: communication between the team members, which considered a crucial factor to successful project management in GSD[30]. The team communication is very difficult in context of GSD, due to the geographic distance that prevents direct communication among team members, which leading to coordination risk. [43].

To allow the integration of Scrum into GSD teams, the need is emerged to develop new practices, or enhance the existing ones. This research introduces a feedback mechanism to be used in Scrum GSD teams, this mechanism is aimed to monitor important productivity factors that influence the productivity of Agile GSD team.

1.1 Background and Motivation

The beginning of the new century witnessed the born of new methodology in Software development which shifted Software development process some levels up, and brought high levels of customer satisfaction compared to the traditional software development methods. Agile methodologies came with principles allowed Software products to be developed in faster rate, with better quality that meets the customers' needs. Agile is based 12 principles that drew the guide lines for the development under this method. Principles like customer satisfaction by continuous delivery of working Software, ability to accept changing requirements, and continuous attention to technical excellence and good design spot light on Agile to be adopted by Software teams that work globally in a setting known as Global Software Development (GSD). GSD is a software development setting where team members are distributed globally with no common physical place existence. The opposite setting is "Collocated team" where team members shared the same physical location or work place. GSD is emerged from the need of Software companies and organizations to enrich the source of skilled developers with low cost, reduce the development time and maximum working hours (follow the sun concept), which mainly enhances the Software quality, and Software teams performance[30]. Teams in GSD setting face challenges; time, physical and cultural dispersion, which result in communication, coordination and control issues among the GSD teams that affect the Software quality and team performance [29, 17]. Agile has been adopted by Software organizations run teams in GSD setting, despite the contradiction between Agile and GSD, for example, one of principles of Agile illustrates that "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation", which looks impossible in teams distributed globally, however, many attempts and successful projects during the time were able to adapt such issues and made use of the benefits in Agile and GSD by applying practices and strategies that made it possible. Agile methodologies are also flexible, and aim to improve the team efficiency as well as the product quality by allowing "inspect and adapt" principle on all of its stages; this principle depends on getting feedback during the different phases of the project on the different components of the Software project: product and process, to enhance them both. However, the practices of feedback about the process focus on the technical perspective in the team, for example in Scrum framework, daily stand up meetings, sprint review and retrospective meetings are feedback mechanisms that either focus on the product, or the process in technical matter. And no feedback is focused on other factors that might affect the productivity.

This research introduces the addition of feedback that focuses on factors directly influence the productivity in Software teams. The feedback on these factors is directed from the team to the management level in the software organization, so that managers and decision-making level to have good insight on what is going inside the teams, which allows to make decisions accordingly to keep maintaining the high level of productivity in the teams.

1.2 Research Questions

This research addresses the following questions:

RQ1: Can integrating feedback mechanism to Agile methodologies affect the team productivity in the GSD setting?

RQ2: what are the influences of the feedback on individual productivity factors, and what its effects on team productivity?

Next chapters discuss the work of addressing these research questions.

Chapter 2

Related work

Agile related topics have been widely researched since it was announced by Agile manifesto[2]. This research is aimed to study the part where Agile integrates with GSD, and the confluence of this on the team productivity in Agile. So this literature review focuses on literature aspects about that. This research defines 3 dimensions from the literature:

1. Agile in GSD: contains the literature topics investigate the integration of Agile in GSD, its aspects, issues and benefits. This allows to discover the current state-of-art about this topic, which allows to define the related issues in applying Agile in GSD.

2. GSD challenges: contains the topics discussing the challenges in applying GSD, which should be the base to start from, to find solution to get rid (or mitigate) these challenges in GSD teams, especially when using Agile.

3. Productivity: Investigates the productivity in Software Engineering teams, which work Agile in GSD and this is important in order to find the factors that are founded to be affecting teams in such setting

	Reference	Agile in GSD	GSD Challenges	Productivity	Feedback
1	[16]		✓		
2	[17]	✓	✓		
3	[30]	✓			
4	[43]	✓			
5	[19]	✓			
6	[15]	✓			
7	[9]	✓			
8	[6]		✓		
9	[10]		✓		
10	[20]		✓		
11	[38]		✓		
12	[26]		✓		
13	[18]		✓		
14	[25]	✓	✓	✓	
15	[24]			✓	
16	[27]			✓	
17	[5]				✓
18	[4]				✓
19	[41]				✓

TABLE 2.1: Literature Review Matrix

4. Feedback: In order to find the work already done, or the strategies that Agile, or GSD teams use to provide solution for productivity in Software development teams.

Knowing about these 4 dimensions, gives a thorough idea about the topic of this research. Productivity is an old topic, started in research since decades. Agile and GSD are relatively new topics compared to productivity in Software teams. This literature review includes topics found in Google Scholar, considering the importance of the topic, and current state-of-art. Table-2.1 shows the 4 dimensions, and references discussed in each one.

2.1 GSD challenges

Considering the benefits of adopting GSD team setting, there are many challenges face teams in such setting, issues in literature are mostly referred to "distance", which caused the following issues:

1. Strategic issues: deciding the work division among the teams in different sites, with different set of experiences and resources, causes this kind of issue. Another aspect of this issue is the resistance of GSD by people working in one site, which might cause less collaboration and more conflicts according to Herbsleb et al[16].

2. Cultural issues: this kind of issues caused by the background differences among the different team members in different locations, issues like miscommunication, and misinterpretation of some words that could be understood as rude by some people with different cultural background [16], this also was referred as socio-cultural issues by Holmstrom et al [17].

3. Inadequate communication: A part of formal communication, where topics like tasks status and responsibilities are discussed among the team members, thre is a different kind of communication, called informal (or corridor discussion) is needed in Software development teams, which supposed to keep the team updated about what is going around them, and what other people are working on. GSD is exposed to the lack of this kind of communication, because of the distance, which causes issues inside the team on different locations [16, 43, 19]. Other aspects of this issue is requirements misunderstanding, cost and effort estimations, risk management, allocation of tasks and lack of coordination [30]

4. Knowledge management: Knowledge in GSD is spread among team members in different sites. Poor documentation and low knowledge about who has specific information also affect the team in GSD [16].

5. Project management issues: When spreading development process among different sites, synchronization among these sites would be critical, in order to avoid rework and misunderstanding about some terminologies among the team [16].

6. Technical issues: This kind of issues could be summarized with the technical limitations, such as network communication latency, which causes inefficiency in code sharing and use [16].

7. Temporal issues: Caused by time differences among teams working on different time zones, this in turn reduces the opportunities for synchronous communication, which considered as the most effective type of communication, increases coordination costs and delaying artifacts delivery in the project [17].

And despite the issues related to the distance, there are desired benefits for working GSD as introduce by Holmstrom et al [17]:

1. Stimulating innovation and sharing best practices

2. Access to rich skill set and various practices

3. Closer proximity to market and utilization of remote skilled work-forces

To mitigate the issues in GSD, Herbsleb et al [16], recommended the well-training and educating the students about working in global teams, besides referring to some tactics and tools for bridging the gap among the teams located in different geographical locations. More studies like Niazi et al[30] identified more issues related to the communication in GSD, such as cost and effort estimations, risk management, allocation of tasks and lack of coordination. They also introduced project management success factors based on Systematic Literature Review (SLR) and questionnaire survey, which are found to be factors of good GSD team management, and found that among 18 identified factors in the study; organizational structure, project managers' skills, communication and collaboration are common factors towards the target of successful GSD team. On the same context. Yange et al [43] investigated the coordination issues in Global Product Development (GPD), and identified the drivers (i.e. the good factors facilitate the coordination), and barriers (i.e. the factors prevents the good coordination) using systematic method proposed at the same research. The drivers were found as "technical communication strength related to features and technical communication between the overlapped processes". Barriers are the geographical dispersion, and temporal distance. Additionally, Handley et al[15] studied coordination and control with respect to the cost in distributed teams, the high control and coordination cost associated with the geographic dispersion as well as the task breadth. More studies elaborated more about these challenges with respect to other aspects such as team performance; Espinosa et al[9] discussed the task and team familiarity and complexity, and how they influence the distributed team performance, besides the known issues, familiarity in team and tasks were founded enhanced the team performance. Kamaruddine et al [19] also highlighted the distance (either in location or time) in their survey study. In addition to the issues listed by the previous studies, they listed more issues related to the distance in GSD: Language difference, lack of face-to-face communication, lack of commitment/team work, low quality communication bandwidth, high communication cost, unprepared communication tools and poor communication infrastructure. All of these issues were raised by surveying global participants in their study.

According to the above, there is a real issue caused mainly by distance differences, folding all the consequences related to the distance (e.g. time, geographical, lack of face-to-face communication, ...etc), and that could dramatically threaten the GSD projects. But considering the promising opportunities, and the added value to GSD, solutions should be available to resolve, or at lease to reduce the effect of these issues on the GSD projects. Regarding the opportionaties of Agile in Software Engineering teams, and its proven ability to increase the customers satisfaction, Agile became a promising approach for GSD, despite the contradiction between the GSD project settings - which mainly deals with teams in different locations, and the basic principles of Agile development that relying mainly on face-to-face communication, Agile could be used in global projects as to be discussed in the next section.

Apparently, due to the issues in GSD caused by distance, the productivity of the team could be highly impacted, and new strategies to avoid, or to mitigate the effect of the distance inside the distributed teams in GSD are needed to increase the productivity.

2.2 Agile in GSD teams

Many companies around the world (ABB, Daimler-Chrysler, Nokia and Motorola)[23] adopted Agile in GSD setting despite the appearing challenges. Many researches investigate this setting and tried to share the experience of companies and organizations adopted this setting.

Cristal et al [6] described some Agile methodologies practices used by a company in running Scrum global teams in 2 pilot projects, the company was not running Agile, to allow practicing Scrum in such setting, the research declared that using Scrum was challenging, not because it is not suitable in GSD setting, but because of the difficulties they faced in changing the mindset of the developers in that company, where Agile was not adopted there. However, Agile showed the ability to be applicable in this global setting, and the company included using Agile in its future plans due to its ability to resolve some global development issues by using different practices in Agile, such as the global task board, which was adapted from the conventional user story board in conventional Scrum. Other recommendations were provided to allow Agile in the global setting, such as documenting the Scrum meeting minutes to bridge the gap between the different team located on different geographical location.

Agile also used to reduce the challenges caused by distance (i.e. temporal, geographical and socio-cultural) among the global teams as discussed in [17], where practicing Agile frameworks; eXtreme Programming (XP) and Scrum, found improving the communication, coordination and control within GSD teams, however, the same research concluded that "no single methodological approach may easily solve these challenges" – referring to distance challenges. On the other hand, Estler et al[10] found that the outcome of GSD project using Agile methodologies has no significant difference of that using structured process (e.g. waterfall) based on a case study on many software projects with distributed teams. Other researches went farther and investigated the success of projects developed by distributed teams; Layman et al [20] found that informal communication between the customer and developer is required in (XP) Agile framework despite the GSD challenges, also found that asynchronous and short communication loops (e.g. email and instant messaging) is a good replacement of synchronous communication (e.g. face-to-face and voice call) in distributed teams, that would allow the good practice of Agile methodolgies in GSD. Systematic literature reviews such as [38, 26, 18] surveyed the literature about the challenges of applying Agile in GSD teams, they also collected the solutions, practices and suggestions for applying Agile in GSD. These reviews also revealed that in order to use Agile in GSD, Agile methodologies should be adapted to work in GSD setting, and that literature is lack of comprehensive framework for working Agile in GSD.

As discussed earlier, Agile is a promising method to be used in GSD due to its feature that could mitigate the GSD issues, however, all the listed studies agreed on the importance of Agile practices in its conventional setting, and they should be adapted to work with GSD projects.

2.3 Productivity Factors

This axis of literature review focuses on productivity side, and its related issue with respect to Software development and Agile methodologies. The main goal of Agile development is to improve the productivity by increasing the throughput, and reduce the cost [33]. However, understanding and putting Agile productivity definitions is not an easy task, because of the lack of research about productivity measurement in Agile methodologies in software development, and they suggested future work to focus on defining the productivity in measurable form, according to Shah et al[36].

Melo et al [25], conducted two case studies in the industry to investigate the Agile team perceptions of factors impacting the productivity, they also investigated which productivity factors were the most adopted by the teams in the study. They found that factors, such as Team composition and allocation, external dependencies and team turnover are said to be factors affecting the productivity. Also found pair programming and team collocation are Agile related practices that affect the productivity based on the programmer expertise and task complexity, they recommend that intermediate and senior developers should not work in pairs to resolve easy tasks, because it is a waste of time. Ramirez et al [31] collected 15 factors that have certain impact on productivity in Agile Software Development, by applying Systematic Mapping Study, on 25 primary studies from the literature, which discussed the productivity in Agile Software Development. These factor; Coordination and leadership, Software Engineering tasks, communication, organizational context, knowledge management tasks, balance of workload, team composition, autonomy restrictions, team cohesiveness, effort of team members, close collaboration, adaptability, external factors, mutual support and mutual trust, were found to have impact on the productivity of Agile Software teams. Later on, the same author elaborated more about the impact of the team maturity on the productivity of Software Agile Development team [33], the team maturity were studied on two dimensions: the Software development process, it included the productivity factors related to the development process (i.e. the members acts to convert the input to output) such as communication and conflict management. And the emergent state (i.e. "attitudes, values, cognitions and motivations that emerge from an individual level and become group-level properties") such as cohesiveness and mutual trust, by using multi-case study research methodology. The research also discussed the correlation among the different factors, and found some high correlation among the elaborated set of productivity factors in each group. These factors are Agile specific productivity factors, that explain the productivity in Agile teams.

On the other side, productivity in software development has been studied intensively for a long time, since the productivity is a major goal for any project, regardless it type (i.e. even it is not Software project), and lots of productivity factors were discovered to have direct impact on the productivity of the team, by reducing, or increasing it. Mohagherghi et al[27] found that reuse in Software development, by creating reusable code, and reuse it many times, increases the productivity, this study, similar to many studies, is based on literature review for collecting the productivity factors investigated during the time. Other factors were known to have impact on productivity were defined in different researches, such as: Team Cohesion, Turnover, Communication, Developer Temperaments, Programmer Capability, Language and Tool Experience, and many others were identified through the literature. Wagner et al [40] have collected 51 productivity factors using a systematic review, collected from the literature since 1975. They grouped the factors into Technical (i.e. factors related to the technical side in the team, such as the used programming language), and Soft (i.e. other factors related to human characteristics, such as the Cohesion in the team).

2.4 Feedback

This topic in the literature review is aimed to discover the current feedback practices in Agile, how they are used and what could be the limitations of them.

Feedback is "information about reactions to a product, a person's performance of a task, etc. which is used as a basis for improvement" [1]. In Agile, fast feedback by customers is one important factor for success, another shape of feedback on development process in Agile is regular incremental builds.[5, 4]. Feedback in Agile known as "inspect-and-adapt" which means "knowing" the current situation and possible changes by applying short feedback loops which

[1]Oxford online dictionary (https://en.oxforddictionaries.com/)

allows Agile to better handle the new changes and conflicts (i.e. adapting according to the inspected conditions)[41]. Beside the iterative nature in Agile methodologies, specifically in Scrum, feedback is known to be used in three locations during the sprint (i.e. the single iteration in Scrum):

1. Daily stand-up meetings: Getting constant feedback about the development process on daily basis, to allow developers staying on track [4]
2. Sprint retrospective: One of the most important practices in Scrum, wherein feedback about the passed scrum from the team members directed to other team members, Scrum master and product owner, which allows the entire team to incrementally improve the development process [4]
3. Sprint review (or demonstration): Feedback about the results of the done work by the customer representative (i.e. product owner). Feedback in this practice is focused on the product itself [34]

Feedback has an important role in Agile methodologies, and different mechanisms are used as discussed earlier. However, these feedback mechanisms focus, and take care about the development process and product delivery, this in turn enhances the customer satisfaction, but do not pay attention to the human factor, which might help enhancing the productivity if more attentions is payed upon factors that impact the productivity of the team. Feedback importance also appears to bridge the communication gap, as well and other issues that are caused because of running Agile on distributed teams. In sake of finding method that keeps the management updated with the status about the development team, and gives them the ability to discover the productivity issues immediately when it is possibly happened, this research proposes a live feedback mechanism, aimed to continuously capture feedback from the development teams, make this visible to the management level. Which in turn will give real insight about what is going on inside the development team with regards to the productivity factors.

From the previous, Agile methodologies are promising in GSD, however, Agile is mainly focus on collocated teams, and was not designed to distributed teams, so the tools and techniques of Agile should be adapted to inquire the GSD.

To the best of the author's knowledge, the literature does not contain explicit methods, techniques or practices that are aimed to enhance the productivity of the GSD teams running Agile. The majority of the researches in the literature were observations and investigations on the industry to investigate the development process itself. And the related challenges, without providing solutions to enhance the productivity.

Starting from the given literature review, and based on the lack of solutions to adapt the productivity in Agile teams working in GSD, this research proposes a solution based on continuous feedback about productivity factors in Agile teams working in GSD, which may enhance the productivity in such teams. The rest of this research discusses the proposed solutions.

Chapter 3

Productivity Feedback Elements

In order to develop a feedback mechanism that depends on productivity factors, it is important to define the productivity factors where the feedback should be given at first place, then to define a method about collecting the feedback on these factors. This chapter defines the productivity factors to be used, as well as how to collect feedback on them.

3.1 Productivity Factors Data Resource

Team productivity is influenced by many factors, this research adopts Wanger's et al study [39] to extract number of productivity factors to use in this research, it is structured review aimed to extract the productivity factors from the literature. It is comprehensive and covers large quantity of references that cover productivity topics in 4 famous research portals: ACM's the guide, IEEE Xplore, ScienceDirect and Google scholar in addition to papers from number of important journals in software engineering for long time period. It also collected these topics using a powerful combination of patterns related to productivity. Wanger collected the only applicable productivity factors in modern software development process and isolated factors the became less important such as "chief programmer team usage" and "previous experience with operational computer". Based on Wanger's survey, 51 productivity factors were defined and grouped into two main categories:

1. Technical factors: Factors related to the technology
2. Soft factors: Human related factors wherein the productivity gets influenced by

The following section discusses the criteria used to select the compatible factors that can be used in context of feedback being discussed in this research.

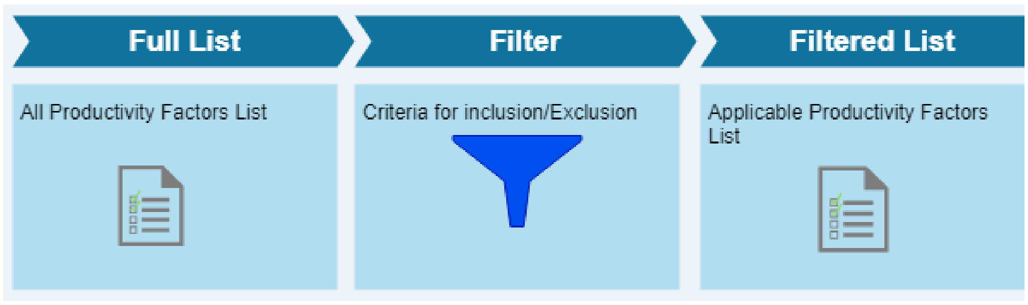
3.2 Selection Criteria of Productivity Factors

There are tens of productivity factors listed in [40], this could be considered as a huge amount of factors, that are very hard to study, also most of them would be unsuitable to the context of feedback and Agile teams, thus these factors should be filtered in order to include the suitable factors, and exclude the unsuitable factors. Figure-3.1 illustrates the selection criteria in action. This section discusses the criteria for selecting productivity factors to be used to collect feedback about.

To include the suitable productivity factors, and exclude the unrelated factors, the following inclusion/exclusion criteria are defined to select among the productivity factors listed in [39]:

1. Measurable: The selected factor should be measurable in order to measure the current level of the productivity factors and provide readable data that reflects the current state of certain factor in the team. Additionally, the way

FIGURE 3.1: Factors Selection


Full List
Filter
Filtered List
All Productivity Factors List
Criteria for inclusion/Exclusion
Applicable Productivity Factors List

the factor should be measured is by asking questions about, the answers should reflect measurable values. Factors that could not be measured by questions/answers will not be included

2. Changeable: The factor should be frequently changeable as the project goes on, to allow getting comparable feedback. For example, factors such like "programming language" and "product complexity" should not be considered since they are unchangeable, so they will be excluded.

3. Context compatible: Some factors are not applicable to be used in the context of this research. The context of this research is providing feedback about productivity factors in GSD Agile teams, so the related topics could be anything related to the Agile team setting, environments or methods being discussed in this research (e.g. Agile and GSD teams). So factors like "Documentation" will be excluded since it contradicts the Agility principles which values the working software over the documentation according to the Agile manifesto[2].

4. Could be self assessed: factor that could be self assessed by team members

based on their experience and participant in the team. Programming language, project architecture, code reuse and other technical factors are examples about technical factors not applicable for getting feedback, hence such factors will be excluded.

Table-3.1 shows the selection of the productivity factors when applying the defined inclusion/exclusion criteria. Factors applicable for the whole set of the defined criteria were accepted, others are rejected to be used in this research.

TABLE 3.1: Productivity factors against the selection criteria

		Selection Criteria			
Factor	Description	Measurable	Frequently changeable	Context compliant	Self assessed
Credibility	Open communication and competent organization	*			
Respect	Opportunities and responsibilities	*			
Fairness	Fairness in compensation and diversity				*
Camaraderie	Social and friendly atmosphere in the team	*			*
Team Identity	The common identity of the team members	*			
Sense of Eliteness	The feeling in the team that they are "superior" and to take pride in the product, team, company etc	*		*	
Clear Goals	How clearly defined are the group goals			*	
Turnover	The amount of change in personnel	*			
Team Cohesion	The cooperativeness of the stakeholders	*	*	*	*
Communication	The degree and efficiency of which in- formation flows in the team	*	*	*	*

Continued on next page

Table 3.1 – *Continued from previous page*

Factor	Description	Selection Criteria			
		Measurable	Frequently changeable	Context compliant	Self assessed
Support for Innovation	To what degree assistance for new ideas is available	*		*	*
Developer Temperaments	The mix of different temperaments on the team		*	*	
Analyst Capability	The skills of the system analyst	*			
Programmer Capability	The skills of the programmer	*			
Applications Experience	The familiarity with the application domain	*	*	*	*
Platform Experience	The familiarity with the hardware and software platform	*	*	*	
Language and Tool Experience	The familiarity with the programming language and tools	*		*	
Manager Capability	The control of the manager over the project	*	*		*

Continued on next page

Table 3.1 – *Continued from previous page*

Factor	Description	Selection Criteria: Measurable	Frequently changeable	Context compliant	Self assessed
Manager Application Experience	The familiarity of the manager with the application	*	*		
Proper Workplace	The suitability of the workplace to do creative work, e.g., windows, natural light, size of room and desk	*			
E-Factor	This environmental factor describes the ratio of uninterrupted hours and body- present hours	*	*		*
Time Fragmentation	The amount of necessary "context switches" of an employee	*	*	*	*
Physical Separation	The team members are distributed over the building or multiple sites	*			
Telecommunication Facilities	Support for work at home, virtual teams, video conferencing with clients	*		*	*
Schedule	The appropriateness of the schedule for the development task	*	*	*	*

Continued on next page

Table 3.1 – *Continued from previous page*

		Selection Criteria			
Factor	Description	Measurable	Frequently changeable	Context compliant	Self assessed
Requirements Stability	The number of requirements changes	*	*	*	*
Average Team Size	Number of people in the team	*			

Stars means the compliance of the related selection criterion in associated column with the listed factor in the associated row. The table does not include the the technical factors listed in [39], and they are all excluded due to the defined inclusion/exclusion criteria.

The next section discusses in details the selected factors that collected 4 starts in table-3.1

3.2.1 Selected Productivity Factors

After applying the inclusion/exclusion criteria on the full list of the productivity factors, the following factors are selected:

3.2.1.1 Team Cohesion

Team cohesion (or cohesiveness) is defined as "dynamic process that is reflected in the tendency for a group to stick together and remain united in the pursuit of its goals and objectives"[28]/page45. Regardless the team setting (i.e. collocated or distributed), Cohesion is important to the productivity of the team [39]. In Agile, where collective code ownership is valuable (i.e individual ownership of code is abandoned), cohesion is mandatory to apply this. Cohesion and high collaboration among team members also enhance code quality and reduce the number of bugs. On the opposite side, low cohesion in the team means "conflicts", conflicts has 2 types: 1) Relationship conflicts and 2) Task conflicts. First type is said to have a negative effect on the productivity, however, the second type has the positive effect[8]. Team conflicts are inevitable, but could be reduced by managing the conflicts, and hence enhancing the team cohesion. The research focuses on the relationship conflicts since it is more compatible with team management rather than task management which should be the responsibility of the entire team rather than the managers. Cohesion in distributed teams is harder than it in collocated ones, it also reported as problem in Agile team [32] which in turn impacts the productivity. Using cohesion as a factor subjected to feedback will give the managers a deep insight about the current conflicts in the team, and allow to enhance the cohesion by resolving conflicts, which consequently enhances the productivity of the team.

3.2.1.2 Communication

Communication is defined as "The degree and efficiency of which information flows in the team" [39]. Communication in Agile team is said to be very important between the team members, as well as between the development team and the users since Agile anticipates change, developers need to have continuous communication with the users of the products (or the customer) to understand the changes, and to take actions [13]. Communication has two complementary types 1) Formal: needs clear, well-understood interface 2) Informal (AKA corridor talks): helps people to keep aware of what is going on. Absence or poorness of communication from both types leads to problems like losing time, misalignment and rework [16]. Unlike collocated teams, in distributed teams, the presence of the second type of communication (i.e. informal) is less due to geographical distance. However, asynchronous communication technologies help in bridging the communication gaps in dispersed teams. The richness of communication is changeable over different phases of the project, and it is needed more intensively in the early development stages of the project. To preserve the benefits of applying Agile in distributed teams, communication should be maintained in all of its forms among the team members. Since the communication is an important factor improves the team productivity, and bad communication affects it, then providing the management with information about the intensity and quality of communication in the team is important to keep up the communication level, and hence to improve the productivity.

3.2.1.3 Applications Experience

Application experience means the level of familiarity with the application domain. This factor is one of the factors that Boehm [3] considered in COCOMO II for cost estimation. It also proven to be important by Banker et al [1] when they analyzed a banking applications with respect to their software maintenance productivity and corresponding influencing factors.

Ideally, level of application experience is relatively low at the beginning of the project, the experience keep promoting as the time goes by. Turnover (i.e. changes in team members) has its impact on the application experience, because of adding new members in the team with no past experience on the application under development. Monitoring application experience level in the project allow the managers to know things about the nature of the project like: the complexity of the project that prevents the growth of experience level, and monitoring the progress of members newly joined the team.

3.2.1.4 Time Fragmentation

It is the amount of necessary context switches of an employee, means that, the time the employee spent in task related to other context rather than the current context of the task the employee is currently working on, or simply could be defined as "Multi tasking".

This factor belongs to a group of productivity-influencing factors called "Teamicide" factors, which are common behaviors organizations take, that kill healthy teams and prevent other teams from reaching a state of high performance [7]. Multitasking could be happened in Scrum for some reasons, for example, [37] reported that the lag time of the product owner to answer the questions raised about some user stories forces multitasking in some cases studied in that research.

Providing feedback about the multitasking in the team looks important to avoid such cases.

3.2.1.5 Schedule

The appropriateness of the schedule for the development task. Inappropriate schedule impacts the productivity and may be caused by: project objectives not fully specified, bad planning and estimating, technology is new to the organization, inadequate or no management methodology, insufficient senior staff on the team, poor performance by suppliers [39], some of these factors are possible to be exist in Agile team. Agile employs practices, such as project velocity that could help determining the over-commitment in project schedule [2]. Appropriate schedule enhances the productivity and quality in software teams. Hence, giving feedback on this factor helps managers good understanding about the schedule and time lines in the team.

3.2.1.6 Requirements Stability

Requirements stability is defined as the number of requirements changes. At the first glance this factor looks contradicting with the Agile manifesto principle (Agile welcomes changes), however, [24] found that very high rate of change in requirements means continuously new requirements, lots of rework, > 30% of the functions new or modified, which obviously impacts the productivity. Lower rate of requirements changes should be acceptable. Providing monitoring on this factors will be helpful for the managers to take proper decision and consequently enhance the team's productivity.

3.3 Productivity

There are many definitions of productivity, however, no clear definition could be found, this is due to few points; Productivity could be found defined as effectiveness, efficiency and performance which leads to misunderstanding and term overload. On the other hand, the measurement of productivity in Software is traditionally defined as the ratio of output, such as lines of code, function points or implemented features, divided by input metric such as time effort. Also software development is counted as mental activity involving knowledge creation, or knowledge use as dominant part of the work, which may change the way of interpreting and understanding the software productivity[25]. This also valid when talking about productivity in Agile teams. In agile, [36] found that lines of code and functional points are surprisingly used to measure the productivity despite the concerns about using such metrics, as they contradict some Agile practices (e.g. code re-factoring – where already implemented code gets edited in better implementation) which ideally reduces the number of lines of code. Additionally, current productivity measurements are not suitable for all of Agile roles (e.g. Scrum master), that is, some Agile roles are not development roles, also the development itself includes other activities are not implying to code creation (e.g. sprint planning in Scrum).

In this research finding a method to calculate the productivity is out of the scope, but a mechanism needed to benchmark the addition of feedback against the team productivity. Sutherland et al [37] made use of Scrum's practice, called team velocity, to determine if moving Agile team from collocated setting into distributed (offshore) setting affects the productivity and quality of the team. In Scrum, team velocity is the amount of work completed in each sprint, calculated by adding the size of completed product backlog item (in story points) by the end of each sprint [34]. Sprint duration is a certain period of time, usually set as 2 weeks to a month [34]. Another example from the professional tools where team velocity is used as team productivity tool is what CA Rally does in reporting the productivity of the team by productivity chart feature [1], this chart feature displays a rate of productivity for last 15 iterations. Rally calculates the productivity rate by dividing the total number of user story points by the available task hour capacity of all team members, this would be even more accurate, since the available time means the actual available time for every team member, excluding vacations, holidays, sick leave, loss or addition of team members.

3.4 Measuring the productivity factors

Different kinds of productivity factors need to be measured in order to reflect numeric readings that managers can understand and react against. The main measurement tool is questions and answers about each one of the selected factors. The structure of questions should lead to close ended answers given to the team members in form of Likert scale answer. The answers will be reflected in form of percentage of how each factor is reported as good or bad in the team. Likert

[1]https://help.rallydev.com/productivity-chart

scale is normally used to collect data from questionnaire, to build a set of possible answers that could accurately reflect facts about the asked questions[22]. It consists of several answers (i.e. Likert items) reflect the level of agreement about the question. Likert scale could come in form of 2-10 possible answers, best practice is to have odd number of answer options to reflect the neutrality options. Answers are given weights, for converting the answers into numeric values that will be used later for analyzing the facts[22]. Table-3.2 shows number of questions and possible respective answers for each productivity factor. Answers to these questions form the measurement for each factor.

TABLE 3.2: Questions and answers for measuring productivity factors

Factor	
Questions	Answers
Cohesion	
1) I can take and resolve any task assigned for other team members	Strongly agree -> Strongly disagree
2) I can help other team members progressing in his/her task	Strongly agree -> Strongly disagree
3) Other team members show interest in helping me in my task	Strongly agree -> Strongly disagree
4) In absence of my colleague, I can work on his/her urgent tasks	Strongly agree -> Strongly disagree
5) Do you feel comfortable to work with your team members	Strongly agree -> Strongly disagree
6) Have you recently received offensive reaction from any of your team members?	Strongly agree -> Strongly disagree
Communication	
1) I receive immediate responses on my queries over email form the other team members:	Never, rarely, sometime, often, very often
2) I receive immediate responses on my queries over email form the other team members:	Never, rarely, sometime, often, very often
3) I communicate with team members using communication media (voice/video calls, chat, ...etc)	Never, rarely, sometime, often, very often
Application Experience	

Continued on next page

Table 3.2 *Continued from previous page*

Factor	
Questions	Answers
1) How do you evaluate the overall experience of the team with regards to the project you are working on?	High, Moderate, Low
2) Do you need significant assistance from the other team members to complete your current task?	Strongly agree -> Strongly disagree
3) Have you recently helped any of your team members who has low experience?	Strongly agree -> Strongly disagree
Time Fragmentation	
1) On how many task you are currently working?	1,2,3 or more
2) Do you feel lack of focus because of the number of tasks you have?	Strongly agree -> Strongly disagree
3) Have you recently stopped working on some task, to work on another ad-hock one?	Strongly agree -> Strongly disagree
4) Are you fully focused on your current task, without interruption?	Strongly agree -> Strongly disagree
Schedule	
1) Do you believe your current task will be completed on time?	Strongly agree -> Strongly disagree
2) Are you waiting for any dependency to complete your task?	Strongly agree -> Strongly disagree
3) Have you recently worked late (and/or) during weekend, to commit to a tight deadline	Strongly agree -> Strongly disagree

Continued on next page

Table 3.2 *Continued from previous page*

Factor	
Questions	Answers
4) Do you feel the estimation of your current task is fair enough?	Strongly agree -> Strongly disagree
Requirements Stability	
1) Is your current task a re-work for old requirements?	Strongly agree -> Strongly disagree
2) How would you characterize your current task?	Strongly agree -> Strongly disagree
3) Have you recently received requirements changes on some of your tasks?	Strongly agree -> Strongly disagree

Questions in Table-3.2 were set by the author to give insights about each factor. Each question is aimed to discover one fact, or more, about the corresponding factor. Factors definitions are mainly used to compose these questions. To reduce the possibility of random answers by the developers, more questions are composed so that they are rephrase about some existing questions, where the answers weights are reversed.

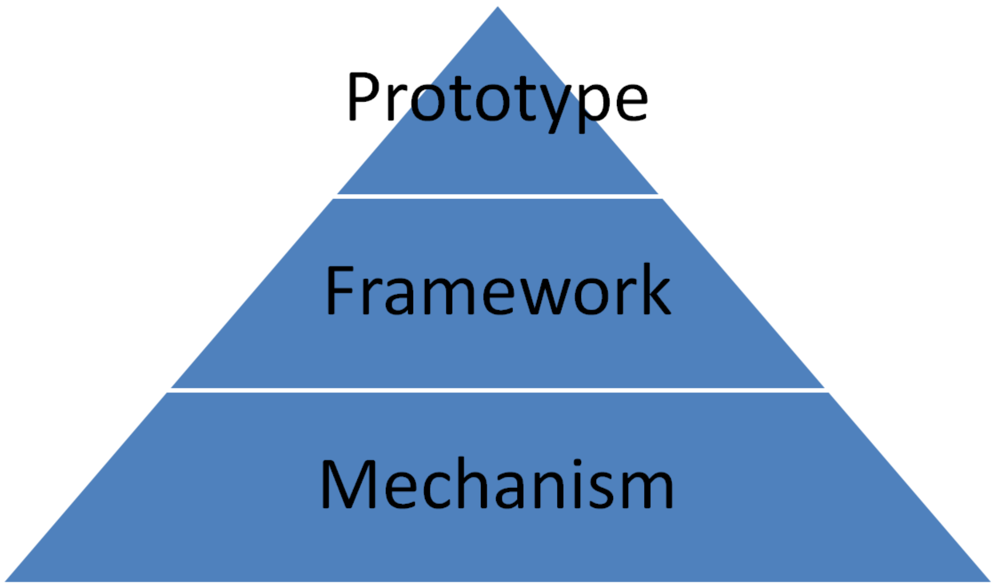
At the end of this chapter, the method of measuring the feedback, and productivity factors were defined, also question pool was created to be used later in this research.

Chapter 4

Live Feedback-based Framework for Productive Distributed Software Development Teams

To provide feedback about productivity factors in Agile Software development GSD teams, it is emerged to define a mechanism to integrate the feedback into such teams, so that the productivity in teams to be monitored and enhanced in the team, in order to make use of Agile in GSD teams despite the challenges and issues could be caused by working in GSD teams. The mechanism is an abstract illustration, that needs to be more specialized, so Framework should be defined. The mechanism could be applied in form of application, or system integrated in companies that use Agile in GSD setting. This research defines the feedback mechanism, after that, proposes feedback framework based on the mechanism. And in order to verify the proposed framework, a prototype is implemented as a proof of concept, to be studied in Agile Software teams working in GSD setting. Figure-4.1 shows the hierarchy of how this research handles the different levels of feedback:

FIGURE 4.1: Live Feedback Levels



- Mechanism: The basement of the pyramid, that discusses the concept of Feedback in abstract way, as well as how to integrate feedback into Agile GSD teams.
- Framework: Is based on the mechanism, gives more details about the implementation of the different items. Framework defines the second level of Feedback implementation, and contains specifications about how could feedback be integrated into teams to meet the defined mechanism.
- Prototype: An implementation of the framework to asses the usage of the mechanism and its effect on the team's productivity. Prototype is one form of implementation of the framework, and could be implemented in different ways, using different technologies.

Next few sections discuss each level in details.

4.1 Feedback Mechanism

Feedback is already introduced in Agile in different ways, and implemented already in Agile frameworks, as in Scrum framework; feedback was introduced in kind of Scrum daily meetings, demonstrations, and retrospective meetings as discussed in the motivation chapter in this research. The type of feedback in Scrum also clarified focusing on the process and does not pay attention directly to the productivity factors in the team, that could critically affect the team's productivity in GSD setting as it empowers the distance differences, which causes productivity implications in such teams. The following points draw the main guidelines the feedback mechanism should have:

- Integration with Agile GSD teams: The feedback should be noticed through out the whole development iteration. In Scrum, the feedback should be provided during the sprint time, in order to monitor the feedback in all Scrum activities. And this would make it different from the feedback activities in Scrum (i.e. Scrum meetings, demonstrations and retrospective meetings). Feedback also should be integrated in a way that minimize the overhead on the team members to give certain feedback about the productivity factors. Figure-4.2 illustrates feedback running during the Sprint life.

- Receiving feedback: Feedback should be received from all participating team members, regardless the location of them, either if they are collocated or distributed. Feedback should be collected on each productivity factor by asking questions on each factor, the answers of these questions should be Likert-based answers in order to minimize the effort on the team members while answering the questions as discussed in Feedback Chapter (Chapter-3).

- Analyzing feedback: Feedback readings should be analyzed in readable format in order to reflect the status of each productivity factors there. Analysis also should be in some visualized form, so that it could be easily interpreted by the managers and remove any possible overhead on the manager to analyze the given feedback values.
- Involving team managers: Managers are the main mentors of feedback, so they have to get access on the feedback provided by team members all the time. Since the feedback could be provided around the clock in GSD teams.

Figure-4.3 shows abstraction on single cycle of feedback. Each cycle starts with getting the feedback from the team members, then analyzing them, after that, displaying the analyzed data to the managers in order to react upon that.

The concept of live feedback could be introduced to development teams by many forms. Feedback could be manually driven; by using face-to-face contact with the team members to get the answers about each questions. However, this could be time consuming, and allow high overhead and cost, since a dedicated resource should be assigned for this job. Another semi-manual method could be used by sending daily online surveys to the teams, which could eliminate the overhead of face-to-face communication and resource allocation, however, live data analysis could be an additional overhead. Automated method could be applied to automatically sending questions, receiving answers and lively analyzing the readings and display them to the targeted level (i.e. management), additionally, due to the distance factor in GSD teams, automation is the best to get feedback from all team members regardless the distance and time differences among team members.

Next sections discuss the framework and the implementation of the framework using a prototype that could be deployed to capture the feedback in real time in the Software development teams.

4.2 Live Feedback Conceptual Framework

This section discusses the live feedback conceptual framework to collect the feedback about the productivity factors from Software teams. Figure-4.4 is the conceptual framework based on the feedback mechanism.

The framework consists of three components:

1. Server: The main component of the framework, responsible to send the questions as messages on certain times, receive the answers by teams, analyzes and saves them. The server is orchestrated by the schedule, which trigger each message post operation, based on certain times during the day. The server is also used to return data to clients as required.
2. Messaging System: This component is an intermediate system, its main purpose is to deliver the messages sent from the server to the subscribed users (i.e. team members). This component also delivers the answers from the client to the server.
3. Client: The component in direct touch with the users, where the messages are received, and their answers are returned back.

Figure 4.5 describes the different actions among the components of the framework. The framework is aimed to deliver the feedback about the productivity factors to the managers. Team members and their manages are the main actors in this framework. The feedback is retrieved by collecting the answers of the team members about a set of questions, posted by Proback to the developers, periodically, during the day, and displays the collected data in readable format to the managers, who in turn can act using the displayed data.

4.3 Feedback Prototype – Proback

Proback, stands for Productivity Feedback, prototype implementation of the framework was discussed in section 4.2, it was implemented using service-oriented architecture, applying Restful web services for connecting among the components of the Proback. The following requirements were set to meet the purpose of the feedback mechanism in Agile GSD teams, and not to add more overhead on the team, especially Proback integrates the Scrum on all of its stages, as daily activity:

1. The system should be able to post questions to team members on intervals during each day.
2. The system should not post questions during weekends
3. The system should show the voting progress, rate, posted question to managers
4. The system should collect and store the answers of the posted questions
5. The system should allow managers to monitor the given feedback in visualized form
6. The system should allow team members to change their answers within the same day of posting the question

7. The system should allow users to answer questions on the same day, and prevent answering questions for past days

8. The system should notify team members who did not answer all of the question before each end of day

9. The system should post random questions, one related to each productivity factor on each day

10. The system should be able to store the answers of team members with time-stamp

It was developed to interact with Slack messaging system, and consists of 4 main components:

1. server: The server contains all of the required API's needed for sending messages and receiving the answers. It also contains the scheduler, which triggers the message postage to the messaging system. The server also contains database, the questions repository, from where the server selects the question to post randomly each time. The last component located on the server, is the WEB server, which reads the database, and displays visual drawings aimed to make the results more readable by the managers. For this part, the server was built on Google Cloud Platform, using Python as development language, and Google Endpoints Framework to create the required API's that either call, or called by the messaging system. Google DataStore was used for data persistence, where the questions, answers, user information are stored in.

2. Messaging system: External system, responsible to deliver the questions to the team members, and send their answers to the main server. Slack was

used as messaging system. It receives WEB Hooks contains question, and corresponding answer set from the server, and deliver back the answers to the server for data storage and analysis.

3. Web Interface: Used mainly by the managers, it displays statistical information about the team answers as well as collaboration level for each team. This WEB site is hoste on the server side itself, written using W3.CSS, Google components, and W3.JS. This interface is responsive site, which means that it could be accessed by either; Desktop or mobile browsers, retaining the same appearance of the components.

4. Client: The application of the messaging system, installed on mobile devices, or accessed by WEB browser, and used by the team members to receive and answer questions. The client could also be any WEB browser used to access the managers view by the managers.

Figure 4.6 shows the components, and interactions among Proback's components from one side, and human-being participants (i.e. developers and managers) on the other side.

4.3.1 Proback's Implementation

This section discusses the implementation of the different components of Proback in details

4.3.1.1 Server Implementation

The main component, hosting the WEB user interface of the managers, and communicates the messaging system for posting the questions, and receiving the answers by the team members.

The server is Google Cloud Platform application [1], written in Python, to provide Google Cloud Endpoints Restful services that is used in different interactions with the other components. The server connects to No-SQL database, where the data of the framework is stored (e.g. questions and answers data, teams info, etc,). The server acts as scheduler for question posting.

In order to avoid overheads on the team members to answer many questions at a time, for that, the scheduler is aimed to distribute the questions posting through over the day.

4.3.1.2 Messaging System

Slack[2] is a web hosted team collaboration system, offers many tools help in team communication and collaboration (e.g. messaging system, voice and video calls, chat rooms, etc). Slack offers API's that could be used for developing special tools and applications run over Slack. Slack users could be grouped in Channels, where messages could be broadcast to channels. Slack also features interactive messages, where the user receive messages with HTML components (e.g. buttons, images, drop-down lists, etc,), which user can reply. Slack received the messages from the Server component, and displays it in a target channel, which includes specific team members. Slack calls API provided by the server, to send the user response back to the server. Figure 4.7 shows sample message received on a slack channel.

4.3.1.3 WEB Interface Implementation

A WEB page, targeted for managers to monitor the feedback received by the team members. Consists of visual WEB components to display the voting data.

[1]https://cloud.google.com
[2]https://slack.com

This WEB page is hosted on the Proback's server. It has 5 different sections, could be navigating among using side menu, as shown in Figure 4.8

1. Overview Section - This section displays the accumulative means for the received answers, for the weighted answers received by the team members. The meter is scaled into 5 steps, from 1 to 5, the lower value is the words, the highest is the best. The reading of the each meter is numerically displayed at the bottom of each meter. The meter also marked in red (from 1 to 2), yellow (from 2 to 3), to indicate the severity of the current reading. This section give indication about the current status of each factor.

2. History View - This view is similar to the first one, but it displays the means of each facts per day, for the past period. From this view, managers can track the history of values, which helps to track the increasing/decreasing of the reading over the time. Managers can select factors from the drop-down list to display history of the selected factor, or select "Overall Status" to display the mean of means of all factors through out the recording period. Data of this section is automatically updated every 5 minutes. Figure 4.10 shows the section, and the selection drop-down.

3. Recent Posted Question - The importance of this view is to give the manager information about the recent posted questions. Information displayed by this view: what was the question, the factor which the question belongs to, the time of posting it, the number of questions sent for the current business day and the time of the next question to be posted, as shown in figure-4.11. Managers can monitor the changes of the corresponding factor on the Overview section.

4. Voting Rate View - Displays the rate of overall voting (i.e. voting on all posted questions) per day during the voting period. The value is the percent ratio from answered questions to total number of posted questions. This view helps the managers to find out the level of the collaboration in a specific team about using Proback. Figure 4.12 shows this view.

5. Team Members Voting Rate - Displays the voting rate per team member. Members with low voting rate (less than 80%> are displayed in red, others (greater than, or equals 80%) are displayed in blue. The rate is calculated by finding the percent ratio between the answers by the team members, and the number of posted questions to the team. Figure 4.13 shows this view.

4.3.2 Question Selection and Posting

Given the 6 factors discussed in section-3, we have a set of questions related to each factor. These questions should be posted to the teams throught the corresponding Slack channels during each business day (i.e. from Sunday to Thursday, from 8:00AM to 5:00PM). Proback has to poll about each factor daily, so that managers can get holistic idea about all of the factors, which also help to track the changes for each factor. To do that, the framework schedule the posting of questions on 6 different times of a day, starting from 9:00AM, and each 90 minutes, to cover all of the factors during full business day. For each time, the framework selects one factor randomly, then selects one of its corresponding question on random way, and post it. Sent question will not be repeated again, when all of the other questions for the same factor are sent, this is to guarantee the posting for all questions related to each factor. The repetition of questions posted to the developers depends on the number of questions for each factor.

For example; if factor X has 4 questions, then the first one will be re-posted on the 5th day. Proback will not post questions after the end of business day, neither during weekend (i.e. Friday and Saturday).

4.3.3 Slack Channels

Each team should have a private channel including the team members. Messages will be posted to each channel, team members can answer the received question on their corresponding channel. Slack channels should be created, and adding the team members by Slack administrator.

4.3.4 Answering Questions

Slack supports WEB access, as well as mobile applications for Android[3] and iOS[4], featuring notification on receiving messages. Developers have to enable notifications on Slack client they use, since it is disabled by default, the user has to enable it. Proback accepts answer by team members only, who are registered in the Proback's database, this is important not to receive answers by non-team members on the Slack channel (e.g. Slack administrators). Answering of questions is by selecting the answer by each team member, the member receives a message after replying each question, that indicates the result of submitting the question. Slack retains the received messages in each channel visible for long time, members are able to navigate through the posted messages. Members can change the answer of any posted questions for the current day only, and disallowed to change past days answers by Proback, this would enhance the validity of the collected answers, by preventing influencing the readings of the factors. Each team member should answer all questions posted per day, to guarantee

[3]https://www.android.com/
[4]https://apple.com

that, email reminder to be sent by end of each business day to each team member who did not answer all of the posted question, requesting to answer the missing questions. Team member could answer the question anytime after the question is received, till the midnight of that day, after that, the answer will be neglected, and the user will receive "Question is expired" message, if he/she tried to answer after the mid-night of each day.

4.3.5 Manager UI

Managers have access to the WEB interface provided by the framework. The WEB page is responsive[5] and cross-browser[6], hence it could be accessed using any WEB browser, on different computer, or mobile platforms.

[5]Responsive web page is adaptive to fit and distribute the UI components, regardless the screen size of the mobile device

[6]Could be used on any WEB browser (e.g. Chrome, Firefox, etc,

FIGURE 4.2: Feedback in Sprint

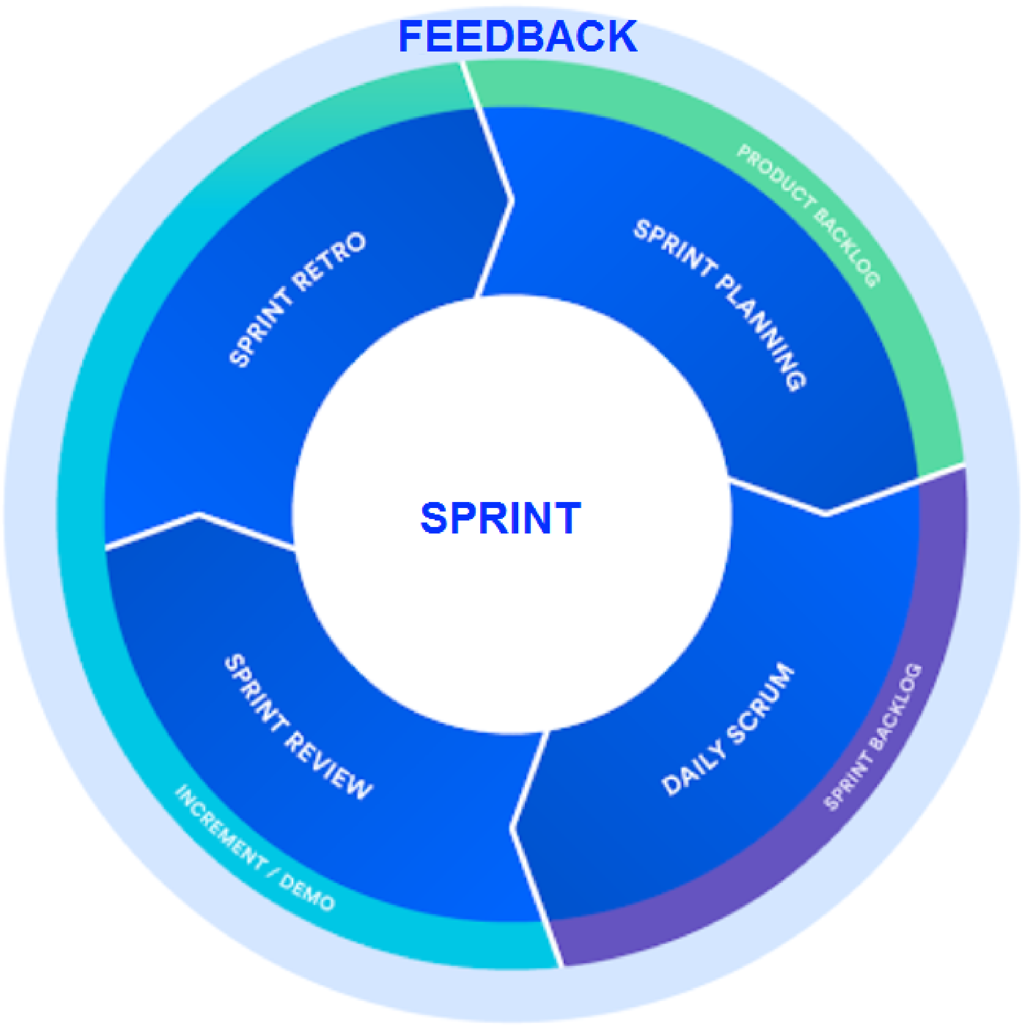


FIGURE 4.3: Feedback Abstraction

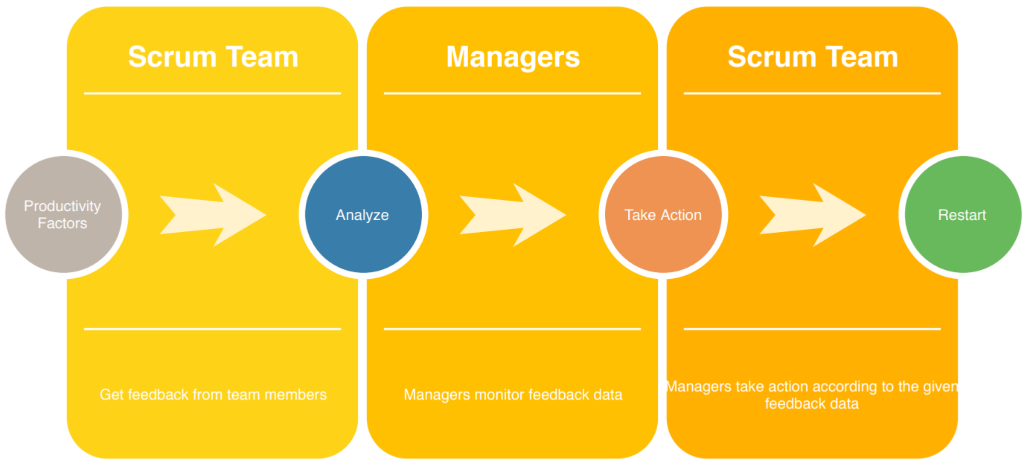

Scrum Team
Managers
Scrum Team
Productivity Factors
Analyze
Take Action
Restart
Get feedback from team members
Managers monitor feedback data
Managers take action according to the given feedback data

FIGURE 4.4: Conceptual Framework

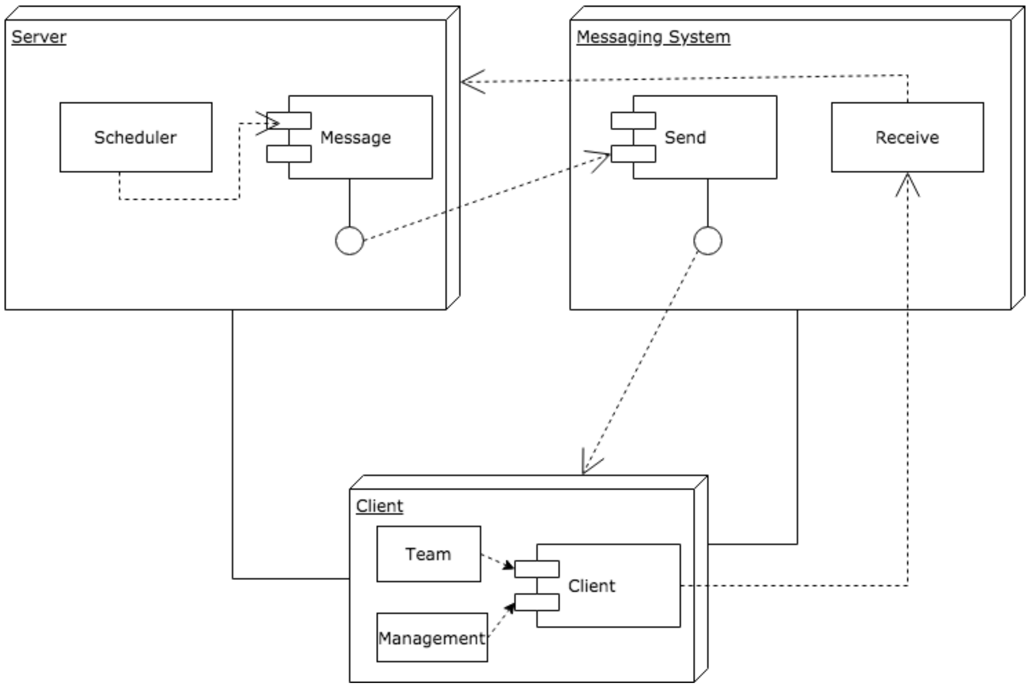


FIGURE 4.5: Framework; sequence diagram

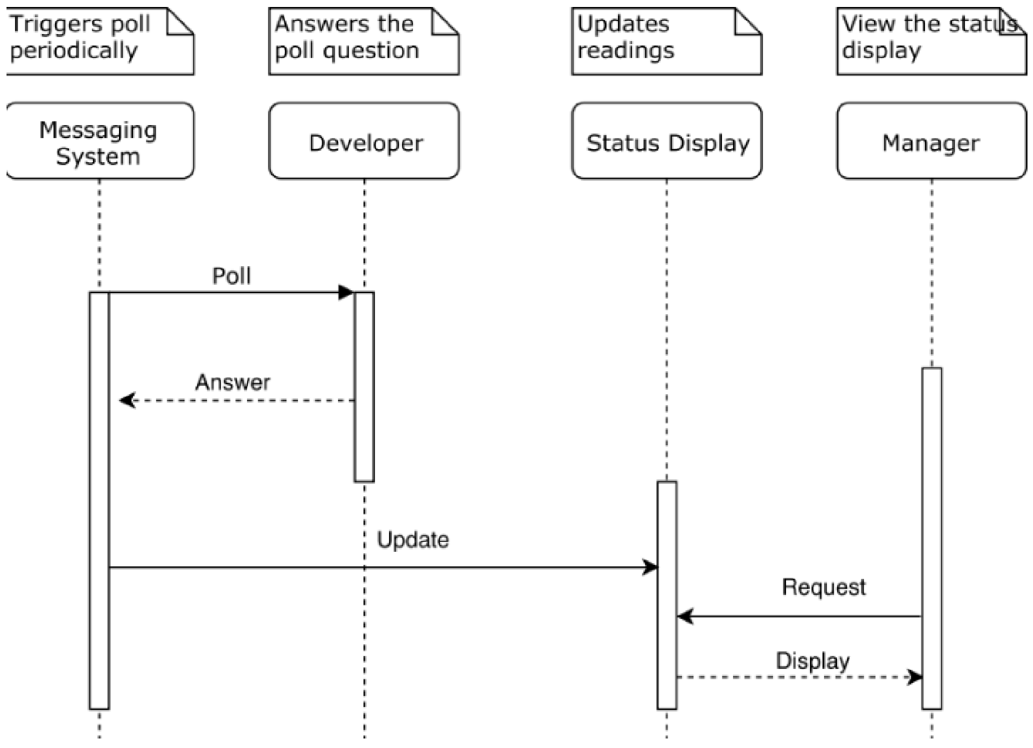


FIGURE 4.6: Proback

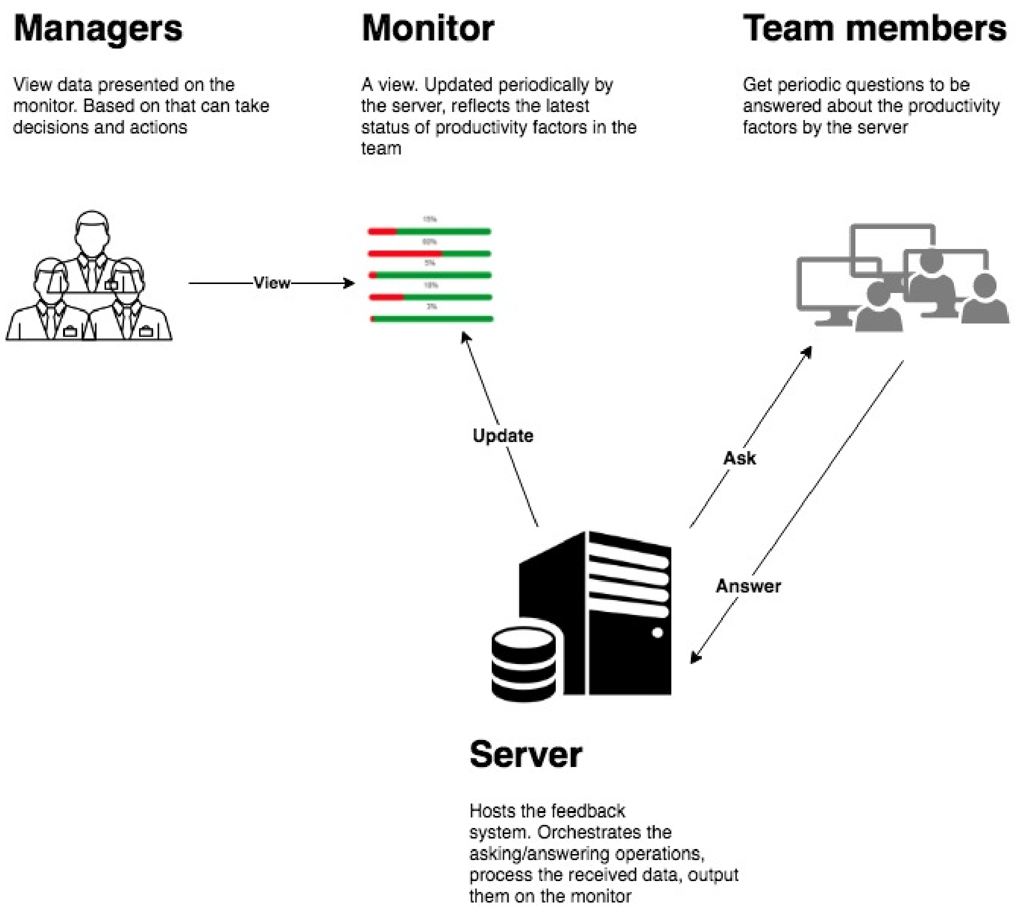
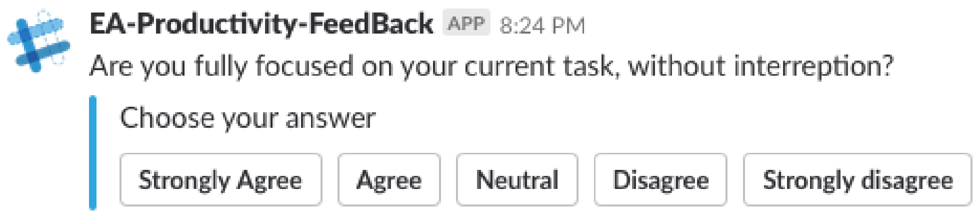

Managers
View data presented on the monitor. Based on that can take decisions and actions
Monitor
A view. Updated periodically by the server, reflects the latest status of productivity factors in the team
Team members
Get periodic questions to be answered about the productivity factors by the server
View
Update
Ask
Answer
Server
Hosts the feedback system. Orchestrates the asking/answering operations, process the received data, output them on the monitor

FIGURE 4.7: Posted Question


EA-Productivity-FeedBack APP 8:24 PM

Are you fully focused on your current task, without interreption?

Choose your answer

FIGURE 4.8: Navigation Menu

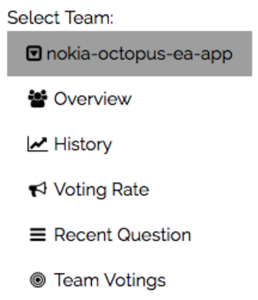
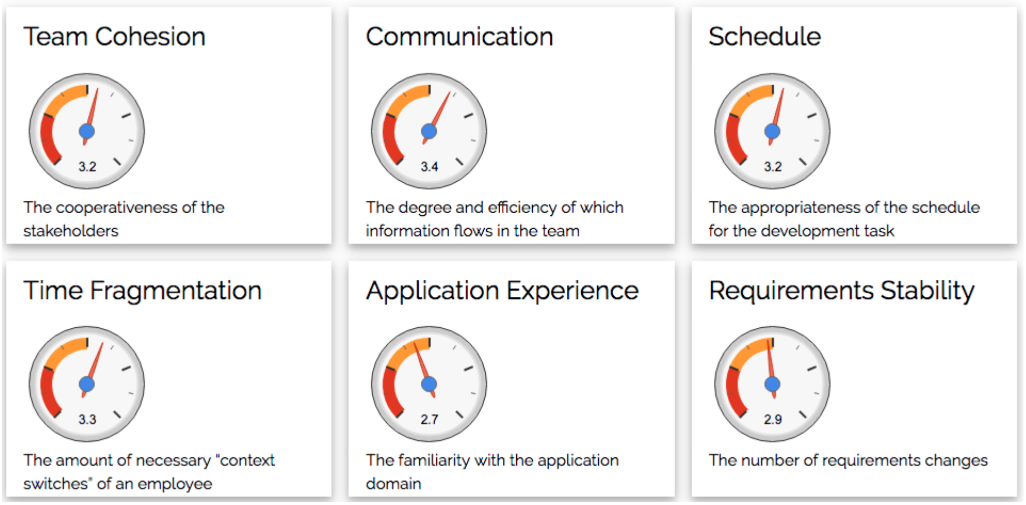


FIGURE 4.9: Overview Section


Team Cohesion
3.2
The cooperativeness of the stakeholders

Communication
3.4
The degree and efficiency of which information flows in the team

Schedule
3.2
The appropriateness of the schedule for the development task

Time Fragmentation
3.3
The amount of necessary "context switches" of an employee

Application Experience
2.7
The familiarity with the application domain

Requirements Stability
2.9
The number of requirements changes

FIGURE 4.10: History View

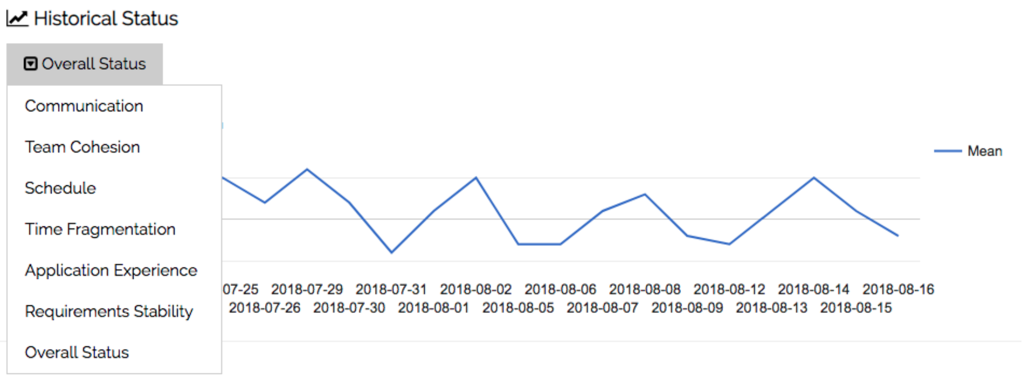
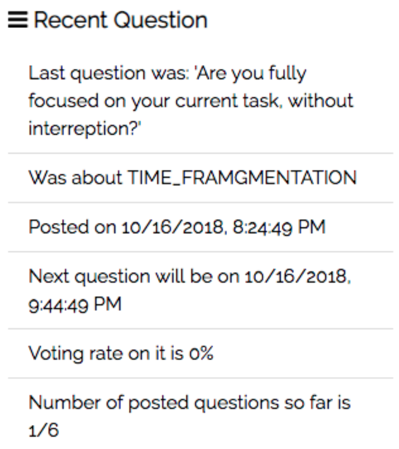


FIGURE 4.11: Last Posted Question Section



Recent Question

Last question was: 'Are you fully focused on your current task, without interreption?'

Was about TIME_FRAMGMENTATION

Posted on 10/16/2018, 8:24:49 PM

Next question will be on 10/16/2018, 9:44:49 PM

Voting rate on it is 0%

Number of posted questions so far is 1/6

FIGURE 4.12: Voting Rate

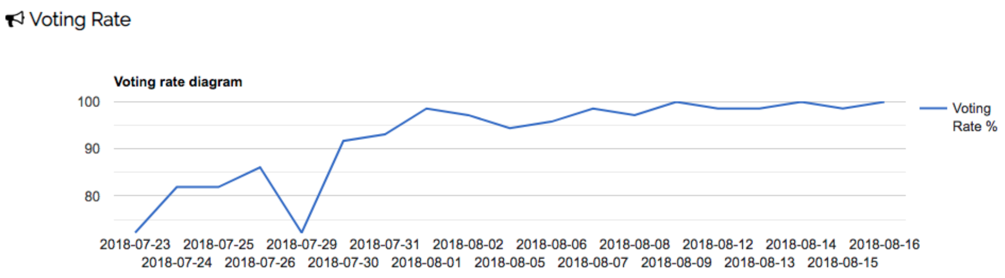
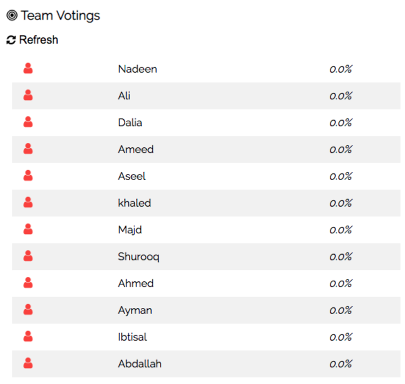


FIGURE 4.13: Team Members Voting Rate



Team Votings

Refresh

👤	Nadeen	0.0%
👤	Ali	0.0%
👤	Dalia	0.0%
👤	Ameed	0.0%
👤	Aseel	0.0%
👤	khaled	0.0%
👤	Majd	0.0%
👤	Shurooq	0.0%
👤	Ahmed	0.0%
👤	Ayman	0.0%
👤	Ibtisal	0.0%
👤	Abdallah	0.0%

Chapter 5

Methodology

This chapter discusses the process of collecting information and data required for the rest of this research. Data could be collected in many ways, based on the nature of the research. This research is an exploratory study, that aimed to explore the effect of adding something new (i.e Feedback) into some existing thing (i.e. Scrum framework). In such kind of researches, among many research methods used in Software Engineering, such as experiments and filed studies, case studies is the most suitable as as preferred by Yin et al [44]. Following sections illustrate the reason of choosing case-study among the other methods, selecting the best fitting type of case-study for this research, design and collect the required data strategies.

5.1 Case Studies

Case study is a qualitative research methodology, normally used for exploratory researches, where the researcher explores one (or more) bounded systems (or case), over time, by collecting data from multiple sources of information (e.g. observations, interviews, surveys, etc.), and reports a case description and case-based themes[44]. According to Yin et al[44], case study could be classified into single, collective or multiple and intrinsic case study, based on the aim of the study. In single use case, a single bounded issue (or case) to be investigated is selected, the boundary of the issue could be individual, individuals, software teams, software applications, etc, through a bounded time. In Multiple cases study is meant to replicate single issue and replicate it on multiple bounded cases. The cases might be selected from several programs in several research sites (e.g. company, organization, etc) , or different programs (e.g. individuals, teams, etc) within single site. These two types of case study are called Instrumental case studies. The third type (i.e. intrinsic case study) is focused on the case itself (e.g. evaluating a program), as the case is influenced by unusual situation [21]. As this research is exploratory about studying the effect of monitoring feedback in GSD teams, this research uses case study as research method to investigate that.

5.2 Case Study Design

Since the effect of adding feedback to Agile GSD teams is being studied using the prototype based on the introduced framework, and it is genuine in the context of team working environments. Therefore intrinsic case study is used to investigate the effect of using the prototype in GSD teams. Also multiple case studies were used to increase the validity or the case study [21].

This case study investigates three different cases, each case represents Software development team, in one Software development local company, where Proback is deployed to be used by each team in the case studies.

Prior to apply the case studies, Proback was developed in section-4.2, to be used in the case study, by the team members and there managers, so that corresponding managers to view statistical information about the given feedback, who in turn can interact towards enhancing these factors in the team.

The questions listed in table 3.2 were used to collect the team members' feedback. The members were asked to answer a set of questions with five-likert scale, that aimed to reflect the state of the each corresponding productivity factor. The results then were visualized and displayed on the managers user interface.

5.3 Case Selection

The case, in case study, could involve an individual, individuals, a program, an event or an activity[21]. The selected cases in this research are software development teams, who are working in Agile Scrum teams, in a distributed work environment. The distributed teams here are stand for GSD teams, since fully GSD teams were not affordable by the company to participate this study. However, most of GSD elements are available in the selected teams:

- Teams are distributed in different locations in the same company
- Teams are participating GSD teams, in different countries. But only local members participate the study
- All teams are working Scrum

The participated teams were consisted of developers and team manager. Team members were all developers, having development task. The selected company is where the author has been working in, that gives better insight about the teams structure, and benefits enhancing the communication between the author and the participating teams. The selection of the teams inside the company was happened based on the requested criteria; distributed Scrum teams. The company is local development company, working on many Software development process, in many development fields, such as mobile, enterprise, test automation, and dev-ops.

Three different case studies are defined as following:

5.3.1 Case 1 - C1

First case study, involved software development team, woks on Enterprise application, since 5 years ago. The application has thousands of users around the world. The team consists of 9 developers, with various experience levels. The team continuously provides new features, and do bug fixing as main tasks during the sprint period of one week each. The manager of the team has more than 15 years of technical and management experience. This case is referred to as C1, T1 to the team and M1 to the manager of this case throughout the rest of this research.

5.3.2 Case 2 - C2

This case is on Dev-Ops development team, working on a well-known Dev-ops framework, since 1 year. The framework has many customers around the world. The team consists of 12 developers, with various levels of experience, and delivering features continuously. The team runs Scrum, with 1 week sprints. The manager of this team has more than 20 years of experience in technical and management fields. This case is referred to as C2, T2 to the team and M2 to the manager in this case throughout the rest of this research.

5.3.3 Case 3 - C3

Software development team, consists of 9 developers, working on stand-alone application, with thousands of users around the world. The team is working on delivering features and bug fixing tasks. The team runs Scrum with 2 weeks sprints. The manager of the team has 12 years of technical and management experience. This case is referred to as C3, T3 to the team and M3 to the manager in this case throughout the rest of this research.

5.4 Data Collection Procedures

The second step of case study is data collection, which is aimed to collect the evidence to be used in the results. Data is collected using several sources to avoid the effect of one interpretation of one single data source [21], which increases the validity of the results. The collected data is used for the next step of the case study (i.e. data analysis).

Data in case-study could be collected from several sources [44]

1. Documentation available from letters, meeting agendas, formal studies, ...etc. In this research this type of source is not available

2. Archival records from the company of the organization where the case-study is targeted to. This type of records could be historical records, such as attendance records in some company. This source is unavailable in this research

3. Interviews, is on e of the most important source of case study information, and it is aimed to have direct information source from the participants of

case-study. This method is used in this research, to understand the managers interactions, as well as the effect of feedback in their teams. Also, another form could be a survey, which is also used with the team members rather than the managers, as it is more efficient to collect data from big number of participates.

4. Direct-Observation on the participants of the case-study in field, to study the behavior of the participants of the case-study. Even though this method is applicable in this research, other methods could be more suitable to this study.

5. Participant-Observation, is aimed the researcher to play a role in the case study, such as a staff member. This also is not applicable to this research.

6. Physical artifacts, this source collects information from sources like technological device, tool or instrument. Due to using Proback's records collected during the study, to investigate the effect of Proback on each productivity factor. This method is used in this research.

Case study data in this research were collected using 3 methods: interviews with managers, physical artifacts (i.e analyzing Proback data) and surveys with the participated team members. In sake of data triangulation as suggested by Yin et al [44], and Lewis et al [21] to increase the validity of the collected data.

The interviews were designed semi-structured, as it is suitable for this kind of case studies as suggested in [14], and were conducted as face-to-face meetings with each team manager (M1, M2 and M3). The purpose of the questions were mainly to understand the managers interactions, and how they affected each productivity factor in the team. And this would give explanation about the behavior of each factor in the team as collected by Proback from the team members. A lists the questions used in the interviews. The managers were interviewed, rather than posting questionnaires, because they are only 3 managers, which makes interview feasible. Also their input are supposed to be very useful to provide explanations about the results of Proback, as they have main role, by interacting with their teams according to the given readings.

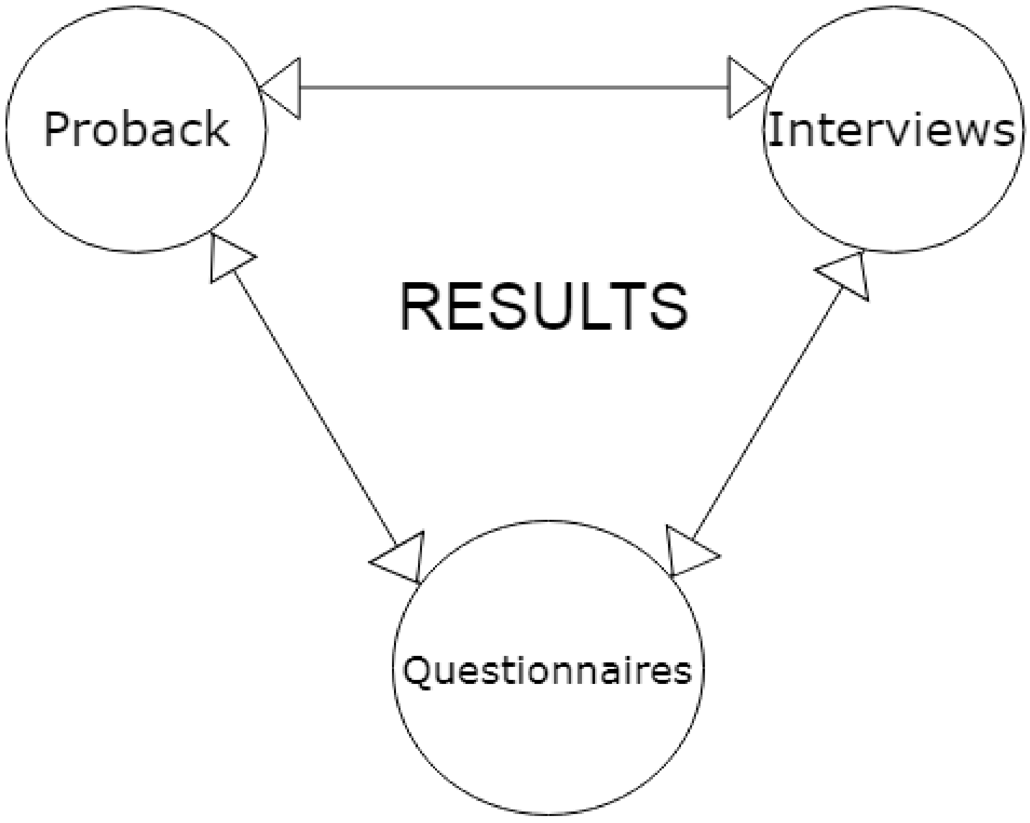
Surveys are aimed to understand the effect of feedback monitoring from team members point of view, as well as collecting information related to the usability of Proback. The usability could be significant for team members to interact with Proback, so that if it is not usable, team members will not interact with it actively.

Proback's database contains the records as received by team members during the case-study period. The result data to be analyzed and related to the answers of the interview questions by managers as well as the survey answers by the team members.

5.5 Analysis Procedures

As recommended in [35, 14, 44], thematic coding is a good method to analyze data in case study research methodology. This method suggests retaining triangulation of data collection and analysis , and maintain the chain of evidence to bring up the conclusions from the collected data [44]. This research follows this method to analyze the data collected from the different sources used. In this method, the interview transcripts and surveys are coded and categorized to derive the final conclusions. The results from each data source is crossed with the other sources, so that the records from Proback will be validated along with the answers of the interviews by managers, as well as the survey questions that are related to the productivity by the team members.

FIGURE 5.1: Data Triangle



To meet the data triangulation, data from 3 sources (i.e. interviews, Proback data and questionnaires) are crossed as shown in figure-5.1 as following:

1. Output from Proback to be used in Questionnaires and Interviews outputs to validate Proback results
2. Output from Interviews to be used in Proback's results and questionnaires to validate the interviews results
3. output from questionnaires to be used in Proback and interviews to validate the questionnaires results

Following chapter displays the results from each data source, also it shows how each result set is analyzed and crossed with the results from other data sources.

Chapter 6

Results

This chapter discusses the results were obtained from the different data sources of the conducted case study discussed in this research; Proback records, interview with managers and the surveys by team members. Next, the results will be crossed connected to each others to meet the purpose of data triangulation.

6.1 Conducting the Case Studies

The company where case studies have applied, has integrated Slack to many in-house information systems, such as human resources, as well as messaging tools affordable to the company's employees for communication. Therefore introducing Slack to the case study participants was smooth, since all the employees were familiar with it, and having accounts on the company's Slack work-space.

Team	Number of members	Sprint length (week)	Number of locations
T1	8	1	3
T2	9	1	3
T3	12	2	4

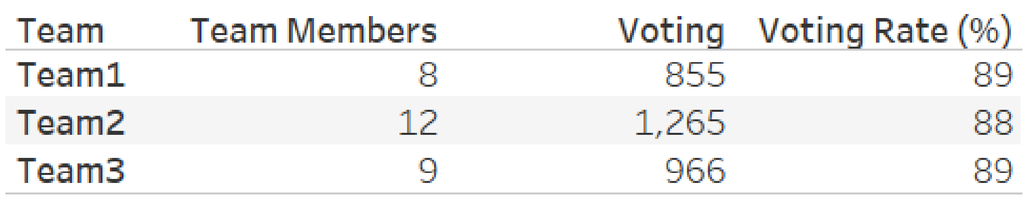
TABLE 6.1: Teams Information

Proback was configured to interact with the company's work-space. The framework was configured to interact with the company's Slack work-space. Three Slack channels were created for each team, added the team members along with their managers in each respective channel and configured on the framework. Proback was loaded with the questions for all factors, and entered members information to be used by the framework. Before starting the study, the instructions were sent via each channel for team members' information (Appendix [B]). For the managers, face-to-face meetings were happened individually with each manager, where the study was discussed in details, and instructions were given about the study. Managers were informed about the importance the study. Confirmation emails were sent to each manager individually (Appendix [C]). Prior to the official run, a pilot run was happened for one week, in order to allow the participants to get familiar with the framework. Data of the pilot run were dropped prior to the official run. Managers were asked not to take actions according the first week's records, and consider this data as the initial recordings that could be acted upon for the next week.

Team members were participated the voting efficiently. As per the records in Proback's database; Proback sent total of (120) questions for each team during the case-study period , and received total (3086) voting records by the members of the 3 teams. Expected full voting records were (3480). Hence the voting rate during the study period was (88.7%). Figure-6.2 shows the number of received questions by each team members. The given voting rate is believed to be valid to get significant results. The missing voting are related to two reasons:

1. Availability of team members: Due to individual vacations for team members and customer site visits by some team members. So absence of team members avoided the voting by them.

TABLE 6.2: Voting per Team



Team	Team Members	Voting	Voting Rate (%)
Team1	8	855	89
Team2	12	1,265	88
Team3	9	966	89

2. Disabled Slack notifications: Slack disables the notifications for Slack channels by default, so instructions were given to enable Slack notifications for the subscribed channel for each team member.

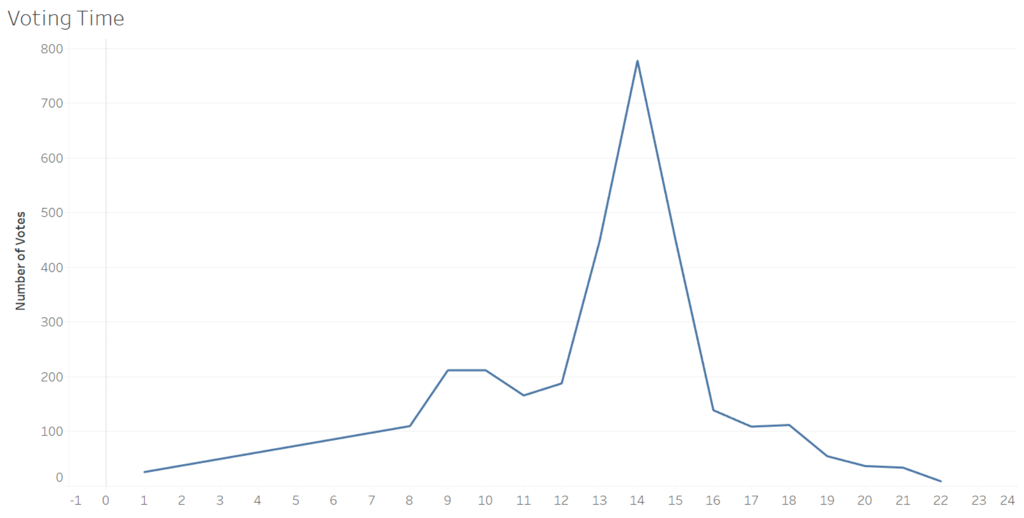
Voting records were received during different time of day during the study period, figure-6.1 shows the peak of voting was around 2:00PM. Some voting records appeared received too early (i.e. around 1:00AM), even the voting was allowed to 12:00AM daily, that was because the timezone on ProtoBack server was set to (UTC), and the timezone for the case-study was (UTC+2), so the lock was actually happened on 2:00AM for voting on the previous day's questions. Team members were encouraged for voting by the researcher, as well as the company management by sending direct messages on each team's Slack channel. Also a reminder email was sent automatically at 3:45PM for each team member having voting rate lower than 100%, this was aimed to have better voting rates.

At the end of the study period, the Proback stopped posting the questions, however, the channels were maintained for later use to send the questionnaires there. Interviews were taken place with each manager (Appendix [A])

6.2 Case Study Database

All artifacts and data given by each data source for each case-study are stored in public location as suggested by Yin et al [44], so that it could be referred to later

FIGURE 6.1: Voting Time



by future studies[1]

6.3 Results from Proback Database

Proback results are sets of data collected by the team members on each team, data are mainly numbers on Likert scale, received as answer on the posted questions for each team. The answers are related to the productivity factor which was the question aimed to measure. Data received were visualized, and provided on Proback dashboard for better readability by the managers, to help them interpret the meaning for each given chart on the dashboard of Proback, showing the date of each day when team members provided their answers. In sake of data analysis, resulted data were re-drawn using a professional data analysis program, called Tableau[2] into more readable format, and categorized into weeks of the case study period; Week1, 2, 3 and 4, where the first week was considered as historical data for the team current situation with regards to the productivity

[1]"https://drive.google.com/open?id=1Syi33hV5$_t hQYytRumimk_a kjWYCtJRZ$"
[2]https://www.tableau.com/

factors in each team. Data in this week were not subjected to managers interactions, which reflects the real situation of each team before adding Proback to the work environment. Given the resulted charts, some results could be drawn before analyzing these results using some statistical measurement, that will be discussed later in this chapter.

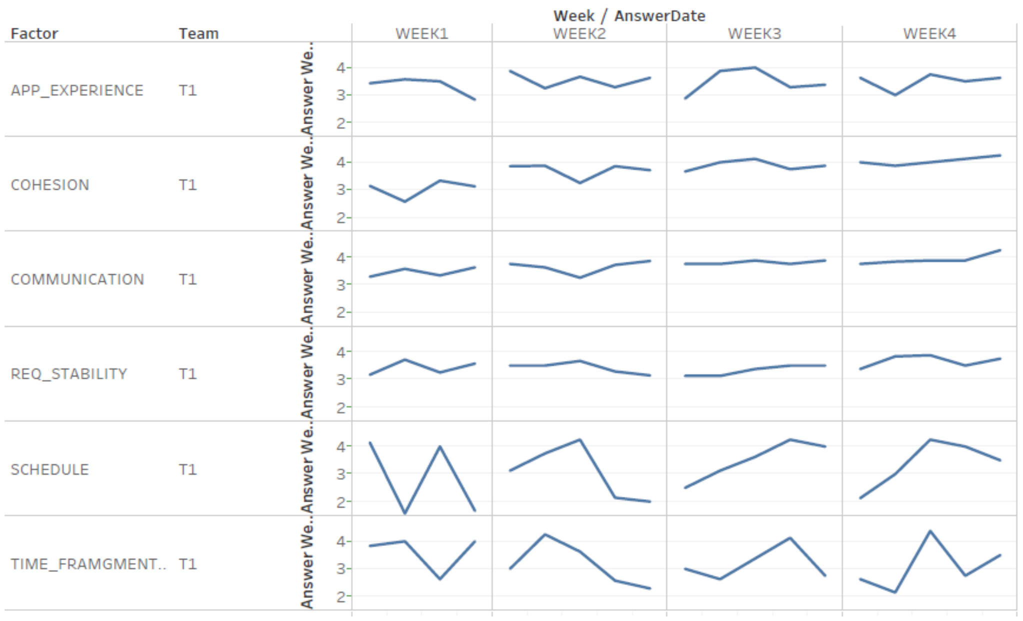
Following subsections discuss the charts drawn using the answers received by each team on each productivity factor used in Proback, with respect to the behavior of the changing values through the weeks of the study. The behavior will later be clarified using the other sources of data; interviews answers, as well as questionnaires results, which would discover if these enhancements are due to the managers' actions, or to other reasons.

6.3.0.1 Factors per Teams Discussion

For each team (T1, T2 and T3), charts show the change over the time of the study. More insight will be given by applying statistical analysis on the drawn data. Charts are important to show the behavior of each factor in each team by the time.

For T1, figure-6.2 shows the voting behavior by T1 on the productivity factors. From this chart, the first week shows, in general, unstable voting values for some productivity factors, mainly Schedule and Time fragmentation. Other factors having more stable results, such as what can be seen in Communication, App Experience and Requirements stability. The chart also shows some increasing values from day to day, especially in Communication and Requirement stability. Following weeks shows more stability on the voting values for most of the productivity factors in general. However, the unstable factors kept

FIGURE 6.2: T1 Voting

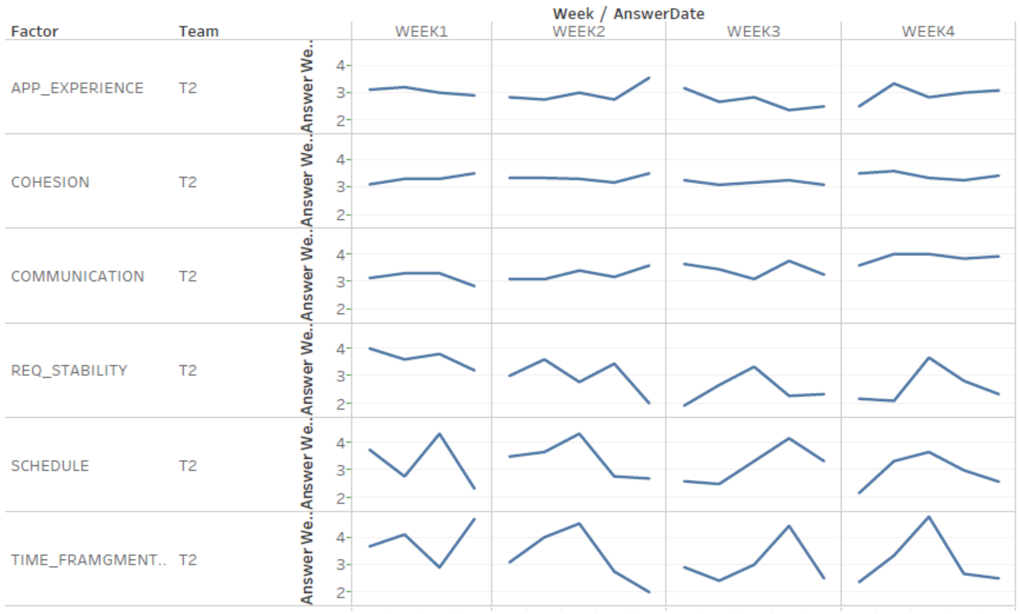


swinging from good to bad, as could be seen in case of Schedule and Time fragmentation factors. Some clear enhancement could be seen in Communication, Team Cohesion, App experience and Requirements stability.

For T2, figure-6.3 shows the resulted data over the time of the study for this team. Similar to T1, Schedule and Time fragmentation look unstable for the first week. Other factors show more stability for the same week for the other factors. However, instability brought clearly for next weeks for other factors like Requirements stability, in addition to Schedule and Time fragmentation. Some enhancements also could be seen for some factors such as the first three: Application experience, Team Cohesion and Communication on weeks 2 to 4. Communication and Schedule show clear enhancement over time compared to the first week.

For the third team, T3, stability on the first week was dominating for most of the factors, however, Requirements stability looks unstable for this week. Also

FIGURE 6.3: T2 Voting

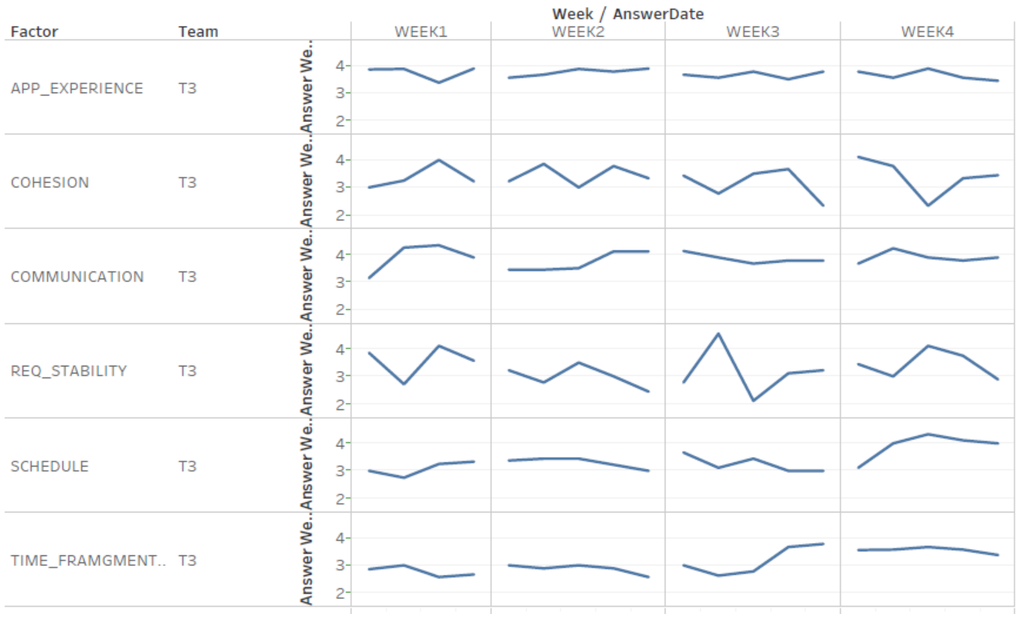


Schedule shown clear weakness, as most of the values are below 3. Other weeks shown some stability for App experience, as well as Communication. Some factors got enhanced with respect to the behavior shown by the chart, for example, Schedule and Time fragmentation.

The readings are also drawn to show by grouping each factor in the teams. Next few paragraphs discuss the characteristic of each productivity factor compared among the teams. Teams properties should be considered in the comparison; the teams are all Scrum teams, working on 2 or more geographical regions, they differ in the projects they are working on, and the managers of them. The characteristics discussed here, will be more elaborated with help of statistical test (Mann-Whitney) in the next section, and will be linked for further elaboration with the answers of the interviews with the managers in following sections.

Figure-6.5 shows the Application Experience as received by each team. this

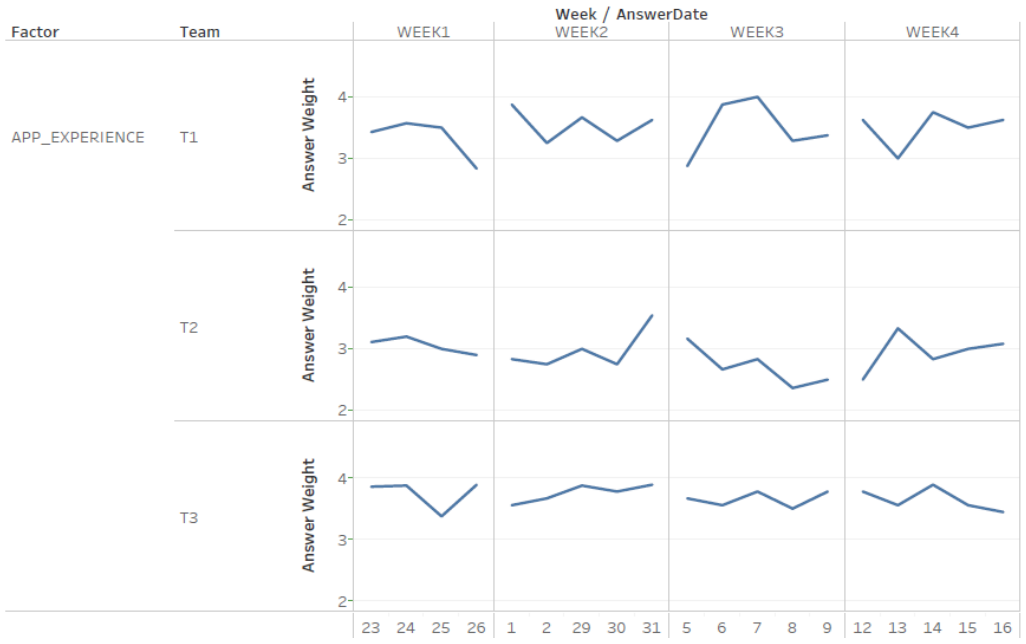
FIGURE 6.4: T3 Voting



factors looks to have unstable characteristic during the study period in all of the teams, however, could be noticed that for T3, the voting has better values than the other teams. Knowing that T3 is newer than the other teams. For T2, values could be seen swinging below 3 in all the weeks. Application experience in nature is gained over time, things like training and knowledge sharing could make it better faster.

Figure-6.6 shows kind of stability for T2, improvement in T1, and instability in T3. To improve the Cohesion, conflicts should be discovered, and solutions should be provided by the managers. Otherwise, this factor will keep getting worst in the team, and threaten it. The characteristic in T3 could tell that there was some conflicts, the manager was not aware about, or the manger knew about it, but he did not take actions to settle the conflict there. Another options is that, the action by the manager did not resolve some issues inside the team, this will be discovered when coming to the analysis part, as well as the

FIGURE 6.5: App Experience Voting

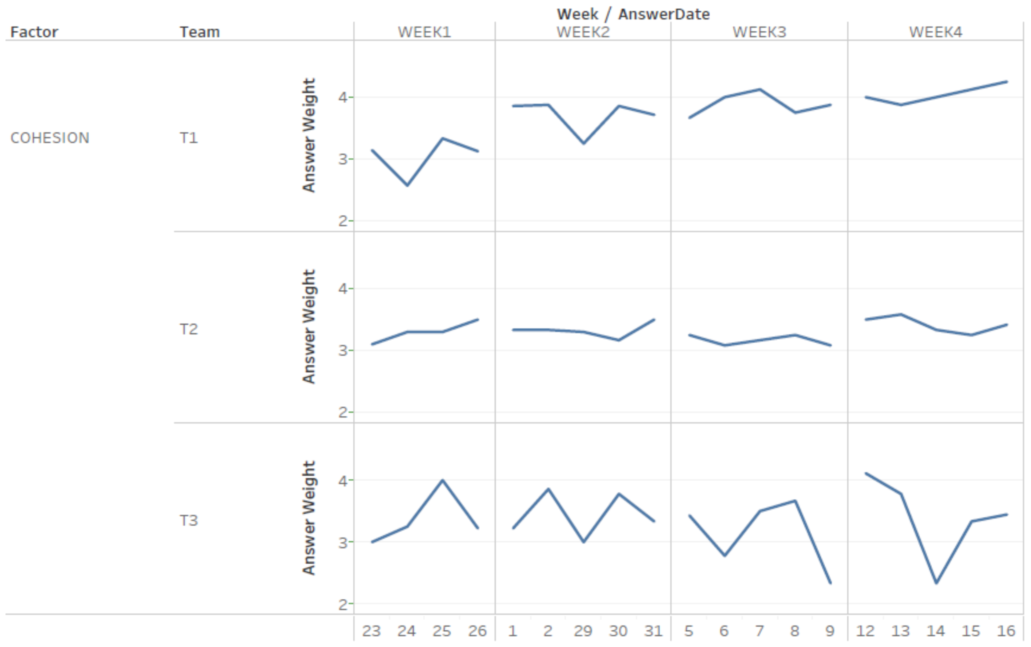


interviews one.

Figure-6.7 shows the characteristics of the Requirement stability in the participated teams. The requirements are said to be unstable at the first stages of the project, and are exposed to changes as Agile welcomes the changes [2], this is clearly seen in T2 and T3, where these projects were relatively new, compared to T1 which was running long time before the other projects.

Figure-6.8 about the voting on Communication by the teams, it shows good level of communication at the first week, and apparently the level of communication looks improved in the weeks later for all teams. The improvement is very clear in T2, the reason will be discovered with help of other data sources, as well as the results of the statistical test to be applied on these values. Communication is vital in GSD teams, and the most important factor that Agile teams are based on, as clarified in the literature review section (section-2). Since all teams

FIGURE 6.6: Team Cohesion Voting

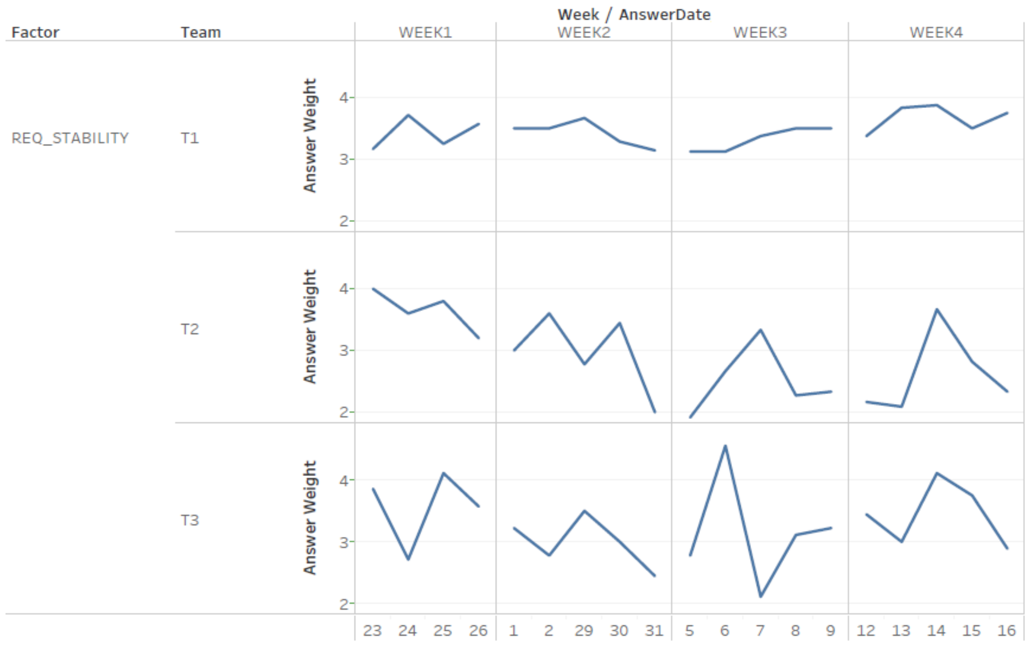


are Agile GSD teams, then it is expected to have Communication in good state. However, the enhancement of this factor as could be seen especially in T1 and T2, could be due to correct action by their managers.

For voting on Schedule by all teams, Figure-6.9 shows instability for T1 and T2, and could not find a special characteristic for them. However T3 shows some improvement in weeks follow to the first one, but the instability could be seen in the 3rd and 4th week. In general, values were swinging above 3, which is an indication that the schedule looks fine in all of the teams. In Scrum, the Schedule is supposed to be fixed, and the team should have defined set of tasks to be done during the Sprint time. This could be the reason about the good readings. The values for T1 will be elaborated more using the other data sources.

And finally, Time Fragmentation voting, as shown in Figure-6.10, shows improvement in T2, rather than the unstable behavior of this factor in T1 and T2.

FIGURE 6.7: Requirements Stability Voting



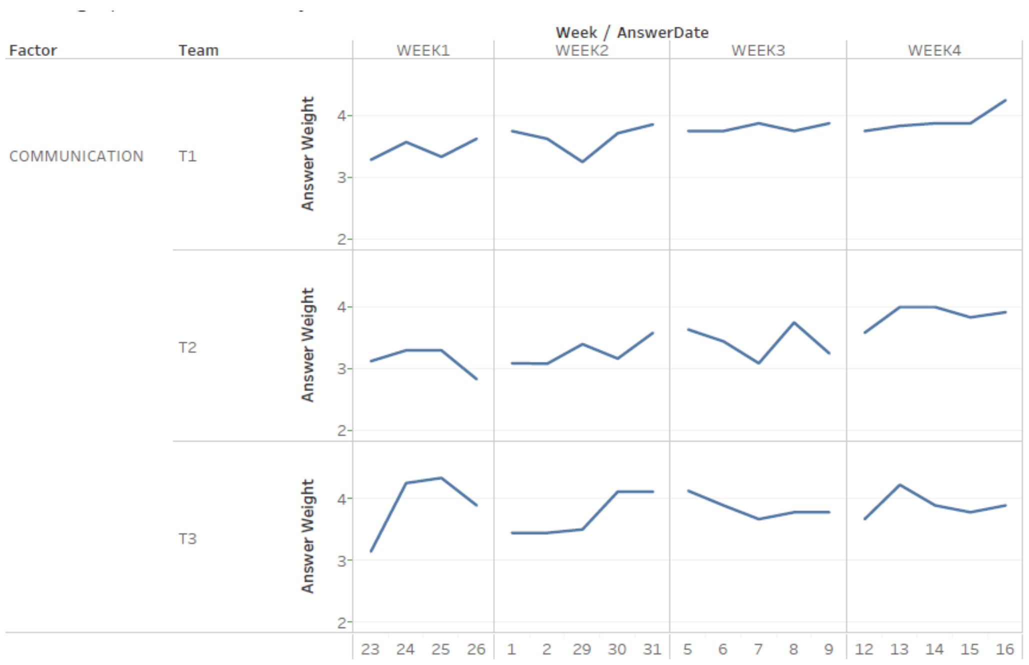
Time Fragmentation prevents team members from focusing on single task, and deliver it on time, and this is said to be productivity threat. Scrum should eliminate such kind of context-switching bu using backlog and grooming sessions.

6.3.0.2 Team Voting Data Analysis

Statistical analysis are important to give more insights on quantitative data, which helps in finding answers on the question the research is aimed to answer. There are many statistical measurements that could be applied based on the nature of the data to be analyzed [42]. Since the factors used in this research are mostly subjective; as they are not exact measurements, and the data may not follow a Gaussian distribution as could be seen in the charts of team voting, then the best measurement to use is Mann-Whitney test.

Mann-Whitney test, is a non parametric test of the null hypothesis that it is

FIGURE 6.8: Communication Voting



equally likely that a randomly selected value from one sample will be less than or greater than a randomly selected value from a second sample[42]. To use this test, Null hypothesis and alternative hypothesis should be defined, the test will either reject, or accepts the null hypothesis. These hypothesis are aimed to find the effect of monitoring the feedback on the productivity factors.

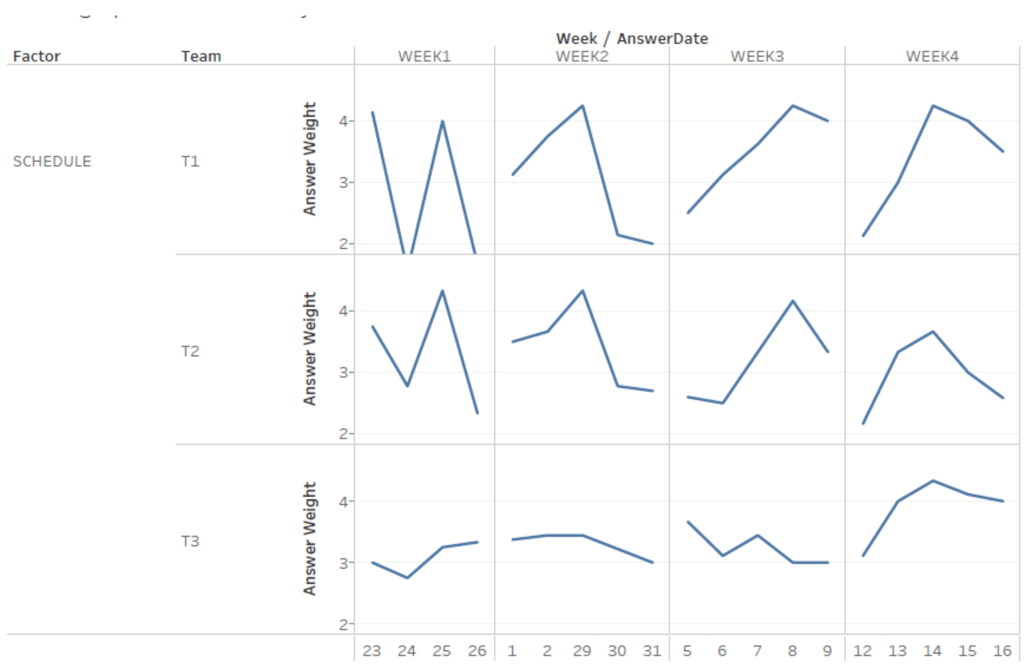
The Null hypothesis (H0) is: There's no effect of getting feedback about certain productivity factor on enhancing that factor in the team.

The alternative hypothesis (H1) is: Adding feedback on certain productivity factor will affect that factor in the team (either impacts or enhances)

Independent variables are defined to be: team size, team experience, the number of locations the team spans over, managers experience, Sprint duration.

The Dependent variable is the Productivity, which is selected in order to conclude the answers of the research questions.

FIGURE 6.9: Schedule Voting



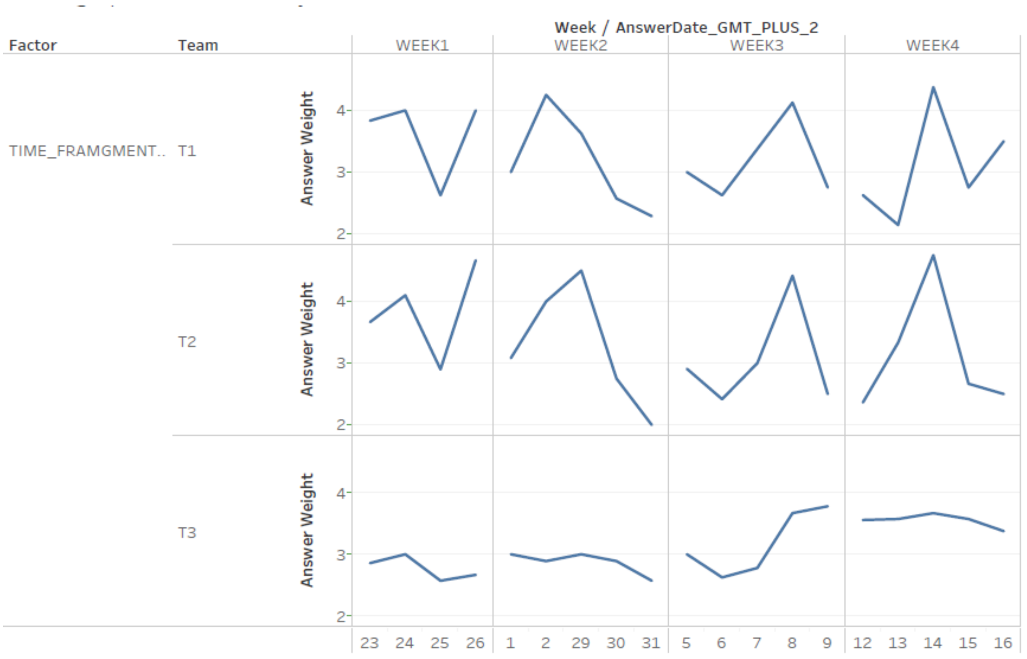
Given the defined hypothesis, applied Mann-Whitney test using R-Studio[3], on Proback data for each team, using 2 groups of data: data given in the first week, against the rest of date in other weeks (Week 2, 3 and 4), which in turn increases the data points entered into the test, and thus increase the statistical strength.

Table-6.3 shows the P-value for each productivity factor in each team collaborated the study. The table shows P-values that could reject the null hypothesis ($p<0.05$):

For T1, N0 has rejected for Cohesion and Communication, which means that for these 2 factors, the addition of feedback has effect on these two factors (H1). Team T2, N0 is rejected for Time Fragmentation and Requirements Stability factors. And for T3, the N0 has rejected only for Time Fragmentation factor.

[3]https://www.rstudio.com/

FIGURE 6.10: Time Fragmentation Voting



Team	Cohesion	Time Frag.	App Experience	Schedule	Comm.	Req Stability
T1	0.0002458	0.1276	0.3204	0.2556	0.04628	0.6185
T2	0.8816	0.003965	0.1509	0.5954	0.05172	0.003959
T3	0.8579	0.02223	0.7622	0.1306	0.0441	0.1966

TABLE 6.3: Mann-Whitney results on the defined data groups

6.4 Results from Interviews

Interviews are the second data source in the chain of evidence in this research. The purpose of interviews is to discover the behavior of the team managers during the study, as well as finding the effect of their interactions on the team, and how that would affect the productivity factors in their respective Scrum teams. This would help clarifying the behaviors of the readings of each productivity factor in each team participated the study as seen from Proback results. The interview also considered to illicit the any suggested enhancements on the managers dashboard, to be used in future works.

The interviews were semi-structured, where questions were not limited to what originally prepared, and the interviews were conducted as discussions with each team manager. Managers interacted with the questions, and discussed aspects around each question, which in some cases, provided answers to different questions in addition to answering the given one.

The questions of the interview were set to ask about the productivity of the teams, before and after applying Proback. On the other hand, other questions were aimed to discover the changes in each productivity factors inside the team. Along with these two aspects, other questions were aimed to understand how did each manager interacted given the data from Proback, in order to link the adaption of each productivity factor with any possible enhancement in the team.

Interview questions are listed in Appendix [A] along with the transcript, coded and categorized[44], in order to analyze the answers in a way follows the chain of evidence[44]. Coding was divided into 2 main catigories:

- Productivity: The main subject to be linked to the first research question (RQ1). In this category, a thorough understanding to be derived about the productivity in each use-case, in the different stages of the study; before, during and after applying the framework in this research. The category also contains the productivity factors, and the related answer for each one. The category is mainly used to justify the data collected using Proback.
- Proback: Is the main tool used to deliver the feedback about the productivity factors. Teams collaboration, managers actions, and factors readings are analyzed in detail to derive answers for both research questions; RQ1

and RQ2. This category contains the answers related to the use of Proback form the managers perspective.

Following few headings discuss each of these categories for each team, connected with each productivity factor.

6.4.1 Productivity related results

The productivity were followed up by the managers of each team involved in the study, however, the rationale for capturing the productivity, and the method of capturing it, were varied among the teams; M1 captures the productivity to track the features delivery progress, as well as using the productivity data for reporting purpose in high management meetings. M1 declared that:

> "I receive weekly task summary from all team members, that allows me to track the delivery of the new features, as well as the defects got resolved during the week. This report is also used by higher management level"

The methods of knowing the productivity in the team were almost same among them, they check the complete vs. incomplete during each spring. This was clarified by M2:

> "We judge the productivity according to the progress of a given task within the given estimation. Delayed tasks are indication of low productivity, we investigate the reason thoroughly to find out the reason of such delays "

All managers (M1, M2 and M3) were satisfied from the level of the productivity before starting the study, as each of them gave rate (from scale of 5) about their

Factor	By Manager
Domain knowledge	M1, M2
Management follow up	M1
Team attitude	M1
Working independently to avoid the affecting the others productivity	M1
Better task estimations	M1
Better task estimations	M1
The knowledge sharing among the team	M1
The setup of the development environment	M2
Turnovers	M2, M3

TABLE 6.4: Productivity Factors by Managers

teams productivity as: 3.5, 3 and 3 for T1, T2 and T3 respectively. M3 declared that:

> ″Although the team members are hard workers, but occasionally we face delays in delivering the task, in most cases, the delays are reasonable″

When managers were asked about the productivity factors, they believe to affect the productivity in the team, based on their past experience, they gave some productivity factors are listed in table-6.4, where each manager added one or more factors they found impacting the productivity of the team based. Although managers answers were anecdotal, the factors they listed are exist in literature as discussed in Chapter-2. Moreover, most of these factors were excluded in this research based on the defined inclusion/exclusion criteria.

For questions related to productivity factors, each manager focused on few factors, rather than the full productivity factors set. Managers justified their interest in some specific factors rather than others. Answers about productivity factors as extracted for each team:

6.4.1.1 Productivity Factors in T1

M1 found that Team Cohesion is low based on the first week's data, after some investigation, he found some conflict between some team members:

> "..I found some conflict between some team members, I had to take action on the management level to resolve the conflict.."

M1 took action, and found enhancement in the following weeks of the study.

More investigations were done by M1, based on the low values in the first week about "Communication" factor. He justified that by:

> "I believe the carelessness by remote team's members caused this. With some investigations, I knew that there are some blocking issues due to this".

Additionally, he found that the big time difference between the teams, affected the communication during the weekends, so some team members had to wait for long time during the weekend of the remote team to get answer about some specific topic. He said:

> "the developer had to wait for the remote team to get back from their weekend"

Schedule was related to Time Fragmentation by M1, and was expected to be low, since:

> "I believe this is due to high time fragmentation to almost all of the team's members. Since the team deliver code into multiple development branches, and have to resolve the issues..."

So M1 accepted this due to the known overhead on the team members appeared in committing changes to multiple locations.

Requirements stability readings were low, M1 suggested doing more communications to understand the requirements before starting the actual work: For the remaining factor; Application Experience, M1 clarified their low levels through the study, because that the team moved to a new development environment:

> "because of moving to a different development environment, to deliver code to other product on the same product line"'

Table-6.5 lists the action taken by M1, on the corresponding rows the table.

6.4.1.2 Productivity Factors in T2

M2, the manager of T2, discussed the characteristics of the productivity factors in his team. For Team Cohesion, he believed that the readings were low at the first week, because the team was new, and his team members were not familiar to each others, also the learning curve of the project were high:

> "team members are almost new to the team, this needs more time for the team members to get more familiar with each others, in addition, the learning curve is a bit high,..."

To enhance this factor, he suggested to actively share knowledge among the team, which in turn would enhance the relationships among the team, as well as enhancing the knowledge there. He believed that this factor was not improved highly during the study, but should be improved through longer time. He said:

Case	Productivity Factor	Action
M1	Team Cohesion	Resolved some conflict between some team members
M1	Communication	Asked the team to escalate any communication issue to the managers of the remote teams
M1	Requirements Stability	Created some agreement to be used during the task requirements elicitation, to reduce the frequency of the requirements changes
M2	Communication	Discussed the communication issues in managerial meeting to act further.
M2	Team Cohesion	Asked the team for more sharing of knowledge
M3	Schedule	Planned to have less task during for the next sprint
M3	Time Fragmentation	Asked the team not to accept any ad-hoc tasks

TABLE 6.5: Actions taken by the managers

> ″...but this won't be happened in near future, as our time is crowded with other development activities″

For communication, similar to T1, M2 related the low level of communication to the communication quality with the remote team members. And he discussed this with the remote team managers, and hence this factor got enhanced in the following weeks, as M2 declared:

> ″...we raised this issue in our managerial meetings, I think that is why it got enhanced.″

Schedule was high from the first week, looking for the reason of that, M2 declared that the first week of the study was a code freeze week, where no code delivery was allowed at that time:

> ″...code was frozen, and no new tasks were taken for couple of days...″

The Time Fragmentation factor, classified as high, and M2 gave a reason why Time Fragmentation could be affected:

> ″'some ad-hoc task could could be taken, and we can not refuse that, since such kind of tasks could important to the customer, and have to be done even if this delays some other tasks″'

Requirements Stability was believed to be low by M2, and he found it strange to get high values for this factor, he declared:

> ″...I remember we had to re-implement full feature as the received requirements were fully changed″'

However, this kind of changes are acceptable by M2:

> ”at least at present, as the project is relatively new, and looking for potential customers that need to be satisfied”

Finally, for Application Experience factor, since the project is new, M2 expected the low level of this factor, and declared that:

> ”As the team is still new, the experiences is still low in this domain. One month would not show big enhancement on the experience.”

6.4.1.3 Productivity Factors in T3

For T3, the inputs from M3 about Team Cohesion, the team members were in high level of harmony, which made the values high starting from the first week. M3 declared that:

> ”The relationships among the team members are good, that leads to good cohesion from the beginning”

M3 raised one issue that could affect the cohesion, which was the high rate of turn-over in the team. He did not take actions on this factor.

M3 was also satisfied with the level of the Communication inside the team, and believed it is vital in the team, so he did not take any action on this factor during the study.

Coming to Schedule, M3 clarified that the time line is always tight, so this factor gained more interest by M3. He acted according to the low level of Schedule values by having less tasks in each sprint, which was aimed to give more flexibility for the team to finish their tasks:

> ”I tried to have less tasks for the next iteration, and that the reason of the good schedule value at the end of the study period”’

Due to side tasks, the Time Fragmentation was low, as per M3:

> ″...found that there are some destruction on the main tasks by some side work for some team members..″

So he acted accordingly by asking developers not to accept any unplanned tasks during the Sprint.

Requirements Stability low values were accepted by M3, as the project is still in its early stages. Unstable requirements were handled by the team members:

> ″we use to receive unclear requirements that need to be clarified. And this all taken care by the team members by proper communications with the product owner″

M3 also commented on Application experience, that the team members are learning rapidly, and related the low level of experience with the high level of turn over in the team:

> ″'Even though the project is new, the experience in the team developed rapidly, however, the high level of turnover makes the experience to be low so far″

From the managers answers, it could be clearly seen that each manager was interested about specific productivity factors rather than the others, and this was varied based on the project properties, for example, new projects, such as in T2 and T3, the focus was on Communication, Time Fragmentation rather than the Requirements Stability, which was always expected to be changed in these 2 new projects. These factors where subjected to interactions by the team managers to enhance them. Also it could be found that Proback gave a thorough insight about the characteristics of each factor in each team to the managers, which enabled them to interact accordingly.

6.5 Results from Questionnaire

The third source of data analysis is the questionnaire. Questionnaire in this research is aimed to input from the team members about the study, with respect to the usability of Proback, where conclusions could be found about how Proback was designed to integrate the Scrum life, on the other hand, it also aimed to find out if Proback affected the productivity of the team members. For this purpose, the questionnaire were designed to ask about the 2 perspective of the required information: team members productivity and Probakc usability. The questionnaire (Appendix D) consists of 8 questions; Q1, Q2 and Q4 are aimed to ask about the usability of Proback. Q2 and Q5 are about how the team found the time they received questions during the study, and if they were disturbed because of that. The remaining questions (Q6 to Q8) were direct question about the productivity as team members can see, and if Proback was a reason of impacting, or enhancing their productivity during the study time.

The questionnaire was sent out to the team members who participated the study, and was opened for answers for one week. At the end of the questionnaire period, 30 answered questionnaires were received.

Divergent stacked bar is used as shown in Figure-6.11 to visualize the answers by the team members, the answers are divided into 2 groups based on the purpose of each question:

6.5.1 Questions Related to Usability Results

According to Figure-6.11 showing the usability group (Q1-Q5):

- Proback usability in general: Q1, Q2 and Q4 show diversions to the agreement about the easy use of answering the questions, disagreement about the need of long time to configure and use Slack, and disagreement about work disturbance because of notifications upon questions receive. Results indicate the user satisfaction about ProBack

- Posting questions: Q3 and Q5; the team members agreed that they were comfortable about receiving questions on intervals through the day, however, the bar of Q5 is diverted more to the agreement about having the questions in one message.

6.5.2 Questions Related to Productivity

Questions in this category were aimed to check the self assessment by the team members about their productivity after using Proback. The questions related to this category are (Q6-Q8). As per the Figure-6.11, all of the answers about these 3 questions were diverted to the agreement side. The team members then found themselves more productive when using Proback.

The divergent stacked bar chart is used here to show the percentage of each answer on each question in the questionnaire clearly, so that readings could be interpreted in correct way.

Taking Q8 for each team reflects the productivity there from the team members' point of view. Since this question (i.e Q8) asks about if Proback was really helped resolving issues in the team, and indeed helped to improve the productivity for each team member. Figure-6.12 shows the answers received from each team about Q8. Team members in the 3 team answered mostly agreement about this question, which means that the productivity was enhanced according to team members' point of view.

From the questionnaire results, it could be seen that Proback is usable by team members, also it did not disturb the team members during their work. And

FIGURE 6.11: Questionnaire, Divergent Stacked Bar

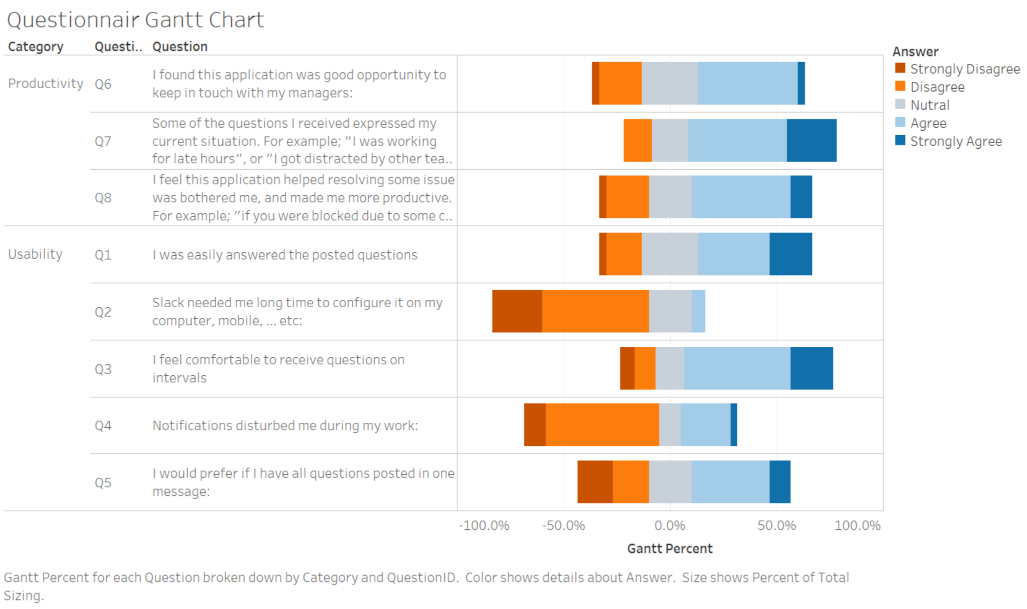
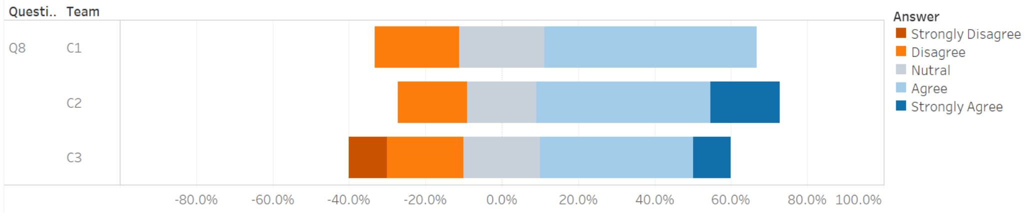


FIGURE 6.12: Q8 for each team



the most important finding is that most of team members agreed that Proback helped enhancing their productivity from their own point of view.

Chapter 7

Discussion

This chapter discusses the results by the case study to answer the research questions. This research discussed multiple factors that have direct implication on the productivity of Agile GSD teams, factors were collected by Wagner et al[39]. To investigate the implications of having feedback on this kind of factors by Software development team members in Agile GSD teams.

6 factors were selected out of 52 factors, based on a define criteria (Table 3.1). This research also presented a framework aimed to collect the feedback from the development team members, to be displayed to their managers, so that managers can take actions to improve these factors in the team. Case study was used as research method, as it is good approach for such research type. To analyze the case study, three different data sources were used: interviews, data collected using Proback and questionnaire, to maintain the triangulation in data, and retain the chain of evidence to raise the validity of this research.

The results were founded in Chapter-6 to be discussed in this chapter, in order to derive answers about the research questions.

7.1 Answering RQ1

Generally, the productivity factors are said to affect the productivity of Agile GSD teams as seen in multiple studies[31, 33, 30, 27]. Recalling the results obtained from Proback; productivity factors were having initial values at the first week in each team, some of them were good, others were not, the changes happened on each productivity factor, in all teams, were varied. Mann-Whitney test on the results were also confirmed enhancements on some productivity factors when applying feedback on them. It could be seen that the N0 was rejected in some cases, which means that adding feedback about productivity factors would affect that factor inside the team. This result was also conformed by the managers as obtained from the interviews, as could be seen in case of M1 with Cohesion, M2 with Time Fragmentation and M3 with Communication. Moreover, supporting these results with the answers obtained by team members about the productivity questions, most of the participants agreed on that to some extent, Proback helped enhancing their productivity. Managers in their answers to Q11 in the interviews confirmed that adding feedback on their teams enhanced the productivity there, this enhancements were varied among the teams.

Providing feedback in each day of Scrum GSD teams could form a kind of overhead on the team members, and disturb them from doing their development tasks, however, the feedback mechanism, as introduced to the teams participated the study in form of Software application (i.e. Proback), took in mind this point, and was designed so that not to add such overhead, and this were confirmed by the answers of the questionnaire by team members.

Enhancing the productivity factors is supposed to enhance the productivity in the team, since the low levels of these factors inside the team could affect the productivity. Hence, the answer of RQ2 would conclude the effect of feedback on each productivity factor used throughout this research.

7.2 Answering RQ2

To answer this question, the three different data sources are to be considered; Proback data showed the characteristic of each productivity factor in each team, also applied Mann-Whitney test to clarify the changes done on each factor from week 2 to 4, assuming the data given in the first week as historical data, where managers did not take actions. The interview and questionnaire questions were designed to explore the changes happened on the factors during the study. Each factor is discussed in terms of the data collected by each source in the following headings:

7.2.1 The Effect of Feedback on Team Cohesion

Team Cohesion is noticed to have different characteristics in each team, starting from T1, the N0 was rejected as (p-value < 0.05), which means that applying feedback on Team Cohesion has a certain effect on this factor in the team. To understand this effect, interview answers by M1 on the productivity factors showed that the Team Cohesion in T1 was improved after applying the feedback, considering the interaction was taken by M1 to improve this factor in the team. In T2 and T1, N0 was not rejected for this factor, which means that applying feedback on those team did not have any effect. This result also conformed by M2, where he elaborated that the team members were new to each others.

Unlike T2, T3 team members were in harmony (as per M3), so level of cohesion started high, and kept high in the team, and M3 did not take any action regarding this.

This result about team cohesion conforms to Mudrack et al [28], where discussed the this factor inside the team, and found that it is vital in teams. However, this study showed that low level of cohesiveness in the team could be acceptable in projects with new joined team members, and could be evolved by time. Even though, as seen in T1, managers can have the ability to enhance this factor with proper interactions.

7.2.2 The Effect of Feedback on Time Fragmentation

The N0 was rejected in 2 teams; T2 and T3. That means that providing feedback affected the Time Fragmentation in those teams. For T2, this factor looks stable from the beginning, its manager (M2) clarified that because there is a kind of focus on the tasks by each team member, so there was no significant overload caused by this factor. M2 did not declare about any taken actions regarding the given readings during the study. Unlike to T2, T3 noticed having low levels of Time Fragmentation in his team, also found the team received more un-planned tasks, which affected their delivery, so he interacted regarding this, the result for the next weeks showed better values. So the Feedback on this factor enhanced it in T3. And finally, for T1, the situation was a bit different from the other teams, where significant time were spent by the team members on delivering the same change on different development repositories, as M1 declared. Looks like the situation was uncontrollable, so M1 did not take action regarding that.

Results showed that Time Fragmentation influences the productivity, as could clearly seen in T1, and this conforms the findings by Demarco et al [7] who declared that Time Fragmentation could prevent the high productivity in Software Teams.

7.2.3 The Effect of Feedback on Application Experience

The N0 for this factor was not rejected, which means whether providing Feedback on Application Experience could affect it in the team or not. Considering the interview answers; for T1, despite the maturity of the team, they were working on new project at the time of the study, hence the Application experience in the new project was still low. T2 and T3, both were relatively new teams, and this factor was low there. M2 pointed to the study period as short, and the level of the application experience could not be significantly improved during this period. All managers did not take actions regarding to this factor, and this could be the reason about why this factor stayed low in the teams. M1 thought this factor could be improved by time.

Boehm et al[3] considered the Application Experience as a main factor in context of cost estimation, however, in Agile GSD, team could be totally new to the application, and this could be acceptable. This study proved that for 3 GSD teams, application experience was not significantly considered. Also, the Application Experience could be changed inside a single team based on the project, as seen in cast of T1.

7.2.4 The Effect of Feedback on Schedule

Schedule results look also not fully affected by feedback. The N0 for this factor was not rejected. Looking at what happened inside the teams; M1 looks mixed between Time Fragmentation and Schedule, and found them highly related to each others, so he dealt with them as one factor. However, N0 was also not rejected for Time Fragmentation as seen earlier in T1. T2, as declared by M2 did not take action, as he believed the level of schedule was good in the team, and since the team runs Scrum, he expected to have the schedule restricted during the Sprint. M3 tried to take action, by reducing the tasks within the Sprint, so that the time line would be better for the team.

From the previous, Schedule looks good in general for Scrum teams, where tasks are defined before each Sprint, and the team focus on these tasks to be done during the Sprint time. This meets the Agile principle about Schedule as announced in [2], and regardless the importance of this factor in Software projects, Agile development in GSD looks generally improved this factor in Software teams.

7.2.5 The Effect of Feedback on Communication

For all teams, N0 was rejected, and accepted the alternative hypothesis (N1), which means that applying feedback on Communication affected the Communication level inside the team. To understand how the feedback affected this factor, each team is investigated using the interviews results. For M1, he found the Communication low at the first week, investigated about the reason and took action regarding that, and found the Communication level was enhanced in later weeks. Similar thing happened by M2 who also took action to improve the Communication in his team. M3 accepted the initial level of the communication, and believed it will enhanced by the time, since the team was relatively new. In general, feedback enhanced the Communication level inside the participated teams.

Communication is vital in Agile GSD teams, especially they are distributed over places. This could be clearly sensed by each manager, and this meets what Wanger el al [39] listed about Communication in Software teams, as well as the results of the empirical study by Green et al [13].

7.2.6 The Effect of Feedback on Requirements Stability

This is the last productivity factor was studied in the participated teams. This factor was affected by feedback in T2, and that could be proved since the N0 was rejected there. However, in T1 and T3, the N1 was accepted instead. According to M1, the team were receiving unclear requirements, and even he took action, the level of requirements stability stayed unaffected as N0 was not rejected in this team. In T2, the case was different; the team was new, and received request to re-implement some features, however, looks the requirements were clear for some extent, hence the level of requirements stability evolved during the study period. In T3, the requirements were changing over the time, and was accepted by M3 because the project was new, and expected to have such requirements stability levels.

Maxwell et al [24] found the level of requirements stability to affect the productivity in the team, however, they look did not consider projects in first stages, and the teams who work in Agile GSD setting, where the requirements stability could be accepted to some extent in sake of meeting the customer satisfaction, as well as the high frequency of unstable requirements at project's early stages, as could be seen in the participated teams.

From the above, the effect of feedback on productivity factors could be varied in each team. Some factors were enhanced, others were not. The enhanced factors were always subjected to managers actions, as could be seen in all of the enhanced factors. Other factors were uncontrolled, so managers could not act against them, so they either enhanced, or stayed at their initial levels. The feedback was seen to have certain enhancement was on: Cohesion, Time Fragmentation, Communication and Requirements Stability. However, Schedule and Application Experience were not proved to have effect on these factor by applying feedback about them.

The results obtained from Proback were validated using the answers of the interviews, as well as the questionnaire answers, and founded to justify the obtained results. The discussion was driven using the different sources of information, which is also aimed to enhance the validity of the results.

Chapter 8

Threads to Validity

Validity should be considered in all of the study phases, to provide robustness and confidence for the results and conclusions of the research [21]. Validity criteria were applied from the beginning on each phase of this study to reach high level of validity as suggested by Yin et al [44], the data validity were met by selecting 3 different source on information.

Runeson et al [35] suggested using 4 levels of validity for research in Software Engineering: construct, internal, external and reliability:

- Construct validity: To what extent the operational studied measures reflect what in the researcher's mind, and what is investigated according to the research questions. For this type of validity, multiple data sources were used; interviews, observation and survey. Data were coded, and analyzed across. The study also supported by a well known, robust statistical test (Mann-Whitney), and considered the risk value as ($p<0.05$). The proposed framework was proved using prototype (Proback), that addressed the features that facilitate the integration of feedback about productivity factors in the whole life of Scrum sprints, in away that would not add extra overhead on team members, and that was clearly seen by the questionnaire answers that Proback did not add more overhead.

- Internal validity: This type of validity is concerned about when examined casual relationship when the effect of some factor unaware by the researcher on the factors being studied. The selected teams were offered by the company where the study was conducted, the teams were all running Scrum and distributed over multiple physical locations, and this conforms distribute setting of the teams running one Agile framework. Another aspect of this type of validity could be seen in the defined questions to ask about the productivity factors. In order to make sure the participants answer the questions as what the author set them about, for each productivity factor, many questions were asked in different ways to ask about same factor, which eliminates the possibility that participants interpreted questions in wrong way rather than what it was aimed to. Questions also repeated over time, and the there was a chance the participant to change the answer during the day, if he/she feels the provided answer was inconvenient.

- External validity: This type of validity is concerned about to which extent the results could be generalized. In case study, the extent is to enable analytic generalization to extend the results to cases with common characteristics by defining a theory. Software engineering theories are still underdeveloped, and case studies naturally have no specified theory [35]. Alternatively, the theory could be based on the literature review. In this research, the literature review resulted in multiple productivity factors, that are used to get feedback upon to investigate the productivity of the Software Engineering teams. This theory is used to drive the case study during this research.

- Reliability: This type of validity is concerned about the ability to replicate the same study by other researchers and get the same results. To conform to this aspect of validity, a case study database was created to contain all the artifacts of the case study; interview transcripts, observation notes, survey answer, in addition to the collected voting by the developers from the Probackś database. Statistical test applied also improved the reliability, as it is a mature and robust test.

Chapter 9

Conclusion

This research was aimed to investigate the effect of the feedback on productivity factors from the team members, on the productivity of distributed teams. Among many productivity factors that have been proven by the literature to have impact on the productivity of the team, we selected a set containing factors meet a defined criteria, considered including the factors that suite to the purpose of this research. Next, a framework was proposed and discussed, aimed to collect the feedback about the selected factors. The framework defined the components and transaction that allow the feedback gathering. After that, a prototype, called Proback, was developed based on the proposed framework. To answer the research questions, case study was selected as the research method, because this research is exploratory in nature. The case study was conducted on industry, by involving local development company takes international projects, and working on distribution team setting. The case study was lasting for 4 weeks, where observations notes were collected as one source of data, the other 2 sources were: interviews and surveys, which have been taken place after the observation period. These 3 sources were maintained over the study to enhance the validity of the research. After that, the collected data were coded and analyzed properly to answer the research questions. The answers of the research questions were derived by the conducted analysis, and showed that the RQ1: Team productivity could be enhanced using the proposed feedback mechanism. And RQ2: Productivity factors could be enhanced using the proposed mechanism by allowing the managers to act given the collected data about the selected factors. However, other factors mentioned in the literature were also proven that they have effect on the overall productivity of the team, the results have highlighted the turn-over rate in the teams. The interviews and surveys also shed light on more features to be added to Proback, that could enhance the feedback mechanism, and consequently enhance the productivity of the team.

For future work, the selection criteria for the productivity factor could be revised to include more productivity factors that could noticeably provide more enhancements to the productivity, such as turn over, which was pointed to by some managers, however, the run period should be longer in order to capture possible changes in such factor. Additionally, Proback could be enhanced to include more features that could enhance the productivity of the team members.

Chapter 10

Future Work

This study was aimed to find the effect of feedback on certain productivity factors in Software teams running in Agile GSD setting. And was more specific to this path. the productivity in Software is a very vast topic, and could be studies in more aspects rather than what was studied in this research. The results of this research covered good opportunities of research related to feedback, as well as productivity factors. Future studies could be more specific to more specific productivity factors, as well as different team setting. Following points summarize some opportunities for future work based on this research:

1. More productivity factors: The productivity factors listed in Table-3.1 was filtered using a defined criteria to make sure selecting the compatible factors with the study context. However, other factors could be selected in future work to discover their effect in different team setting rather than the one selected in this study.

2. Application enhancements and automation: Proback was a prototype to validate the feedback framework studied in this research. It could be enhanced in many aspects; UI, questions customization and posting options. For UI, the managers suggested some enhancements that could be added

there, these enhancements could be found from the managers answers about Question 9 in Appendix-A. For question posting part, the posting of questions were unified for all teams, however, the results showed difference of interest between each team, so question customization based on the team would give better results.

3. Voting automation: Instead of asking participants to answer on certain question, the application could be designed to conclude the answer automatically, for example, instead of asking about requirements stability, the application could be linked to the defect tracking system used by the team. Other questions could be answered by checking the availability of the team member during the day, by checking about its status, or by linking the application with the attendance system of the company.

4. Backward feedback: Instead of polling team members for feedback, allow the team members to give his own feedback about something happened inside the team.

5. Other Agile Frameworks: The team participated in this research were all running Scrum, it worth to have a different study on different Agile framework, such as eXtreme Programming.

Appendix A

Interviews

A.1 Interviews Questions

Symbol	Question	Related RQ	Category
Q1	Do you measure the productivity to your team occasionally?	RQ1	PRODUCTIVITY
Q2	What is the measure you used to judge the team productivity?	RQ1	PRODUCTIVITY
Q3	On a scale from 1 to 5, how was your team productive, based on the team records for the period before using the system?	RQ1	PRODUCTIVITY
Q4	From your own experience, what do you think the reason your team have this level of productivity?	RQ1	PRODUCTIVITY
Q5	Why do you think this factor has low/high initial value (i.e. for the first week)?	RQ1, RQ2	PRODUCTIVITY, FRAMEWORK
Q6	From week to week, why do you think this factor: enhanced, weakened over the time (i.e. for the first week)?	RQ1, RQ2	PRODUCTIVITY, FRAMEWORK
Q7	Did you take any action, after the first week, mean to enhance any factor based on the reading?	RQ1, RQ2	PRODUCTIVITY, FRAMEWORK
Q8	How do see the UI of the factor monitor with regards to its usability?	RQ2	FRAMEWORK
Q9	Do you feel the provided data were enough to take action?	RQ2	FRAMEWORK

Q10	What do you suggest to enhance the tool for better use?	RQ2	FRAMEWORK
Q11	On a scale from 1 to 5, how became your team productive, based on the team records for the period before using the system?	RQ1, RQ2	PRODUCTIVITY, FRAMEWORK
Q12	Based on team's record, do you observe enhancement in team's productivity after using the system?	RQ1, RQ2	PRODUCTIVITY, FRAMEWORK
Q13	Do you feel knowing about such factors is good/bad for you to discover your team more deeply?	RQ1, RQ2	PRODUCTIVITY, FRAMEWORK
Q14	Were you already knew about the status of the factors being fed back by this system?	RQ1	PRODUCTIVITY

TABLE A.1: Interview Questions

A.2 Interviews Transcripts

A.2.1 Answers by Manager1 (M1) for Team1 (T1):

A.2.1.1 Answer on Q1:

It is required to track the productivity sometimes, maybe monthly. [**RQ1, PRODUCTIVITY**]

A.2.1.2 Answer on Q2:

I receive weekly task summary from all team members, that allows me to track the delivery of the new features, as well as the defects got resolved during the week. This report is also used by higher management level. [**RQ1, PRODUCTIVITY**]

A.2.1.3 Answer on Q3:

It is stable, 3.5/5. The team is mature, since we are working on this project since 5 years. I remember the very beginning of the project, we ran into many productivity issues, which required prompt tracking of the productivity continuously. [**RQ1, PRODUCTIVITY**]

A.2.1.4 Answer on Q4:

That would be due to many reasons, I could say that due to: [**RQ1, PRODUCTIVITY**]

- The domain knowledge increased over time (experience)
- The good management follow up

- The team attitude, all know their responsibility
- Working independently by the team members, which means tot affecting the others productivity
- The better estimations for tasks, which were evolved through the time
- The knowledge sharing among the team

A.2.1.5 Answer on Q5, Q6 and Q7:

- Team cohesion: Initially was low, based on the reading of the first week, I found some conflict between some team members, I had to take action on the management level to resolve the conflict. So I had a discussion with the conflicting parties about the impact of their actions on the overall team stability and productivity, and conveyed the message that further action could be taken for the sake of the team's well-being. [**RQ1, RQ2, PRODUCTIVITY**]
- Communication: Initial readings were low, after the investigation, I found that was related to communications with other remote teams. I believe the carelessness by remote team's members caused this. With some investigations, I knew that there is some blocking issues due to this, so I asked those team members to escalate the blocker to the manager of the blocking parties. Another case was happened due to weekend time difference to our team, as well as the big time difference with another team (9 hours behind our time), the developer had to wait for the remote team to get back from their weekend. [**RQ1, RQ2, PRODUCTIVITY**]
- Schedule/time fragmentation: I think both schedule and time fragmentation are related to each others, initial values are low, and kept low. I believe

this is due to high time fragmentation to almost all of the team's members. Since the team deliver code into multiple development branches, and have to resolve the issues, and handle special cases for each development branch. That allowed high rate of context switching. At the same time, schedule is very strict, developers are usually work for late hours, and during weekends to meet the deadlines. This team is dependable on by other teams, and multiple projects. (multiple factors affect the project)... dependencies are also high on other parties, which might block the progress. **[RQ1, RQ2, PRODUCTIVITY]**

- Requirements Stability: Was low initially, and improved over the time. For some case, where the developers complained about the unclear and changing requirements, I advised for more communication with other stakeholders to understand the requirements better. So requirements became more mature. Also the team created some contract for requirements elicitation, to be agreed upon with all stakeholder, this also helped to reduce the ambiguity of the requirements. **[RQ1, RQ2, PRODUCTIVITY]**

- Application Experience: The low initial value is because of moving to a different development environment, to deliver code to other product on the same product line. I believe this should be evolved as time passing. **[RQ1, RQ2, PRODUCTIVITY]**

A.2.1.6 Answer on Q8:

Usable, as landing page, with description on meters, gave the basic idea to understand the overall picture. I used them to link between the readings, and the subject of them. **[RQ1, Future Work]**

A.2.1.7 Answer on Q9:

It Depends on the factor… for example, for team cohesion (vs. time frags), no need to know more information about that meter (mostly visible). But for example: application experience, is not visible, and manager needs more information from the individuals to dig through things more (action needs to be taken on individual)... think about training (all vs individual). **[RQ1, Future Work]**

A.2.1.8 Answer on Q10:

A view to add more questions by managers, I think managers might need to add questions to ask about specific case happened in the team. **[RQ1, Future Work]**

A.2.1.9 Answer on Q11:

4/5, I realized that by the weekly productivity reports I receive from the team members. **[RQ1, PRODUCTIVITY]**

A.2.1.10 Answer on Q12:

Yes, slightly low improvement, but I think that would be enhanced more if we run the system for longer time period. **[RQ1, PRODUCTIVITY]**

A.2.1.11 Answer on Q13:

Yes, I think it give good knowledge about what is happening in the team, especially such kind of factors are hidden.

A.2.1.12 Answer on Q14:

Yes for some of them, like schedule, as I knew some team members were working in weekends. Others no, like Cohesion, I only knew about some conflict inside my team when used this application

A.2.2 Answers by Manager2 (M2) for Team2 (T2):

A.2.2.1 Answer on Q1:

Yes, since the project is still new, we follow up closely with the team to make sure they progressing well. We discuss the productivity in some managerial meetings as well. **[RQ1, PRODUCTIVITY]**

A.2.2.2 Answer on Q2:

Basically, and because of unclear and ad-hoc requirements, we judge the productivity according to the progress of a given task within the given estimation. Delayed tasks are indication of low productivity, we investigate the reason thoroughly to find out the reason of such delays. **[RQ1, PRODUCTIVITY]**

A.2.2.3 Answer on Q3:

About 3/5, this is due to multiple reasons, mostly the given estimations given to meet deadlines. Other reason was due to the dependency with other teams in different locations. **[RQ1, PRODUCTIVITY]**

A.2.2.4 Answer on Q4:

I think it is related to: **[RQ1, PRODUCTIVITY]**

- The domain we work on, it is new to the team, which means new experience, new issues that are not discussed on the Internet communities
- Development environment setup, it is a complex setup that takes long time to get it ready for development
- High rate of turn-overs, many developers left the team, and others came new, which means new training, and hence, more destructions to the other team members.

A.2.2.5 Answer on Q5, Q6 and Q7:

- Team Cohesion: Was low, team members are almost new to the team, this needs more time for the team members to get more familiar with each others, in addition, the learning curve is a bit high, which makes it hard that everyone knows everything in the application's field. Seems not improved highly, however, I think this will be enhanced by the time, I suggested doing knowledge transfer sessions among the entire team, but this won't be happened in near future, as our time is crowded with other development activities. **[RQ1, RQ2, PRODUCTIVITY]**
- Communication: Was relatively low, I think this is due to the communication level with the remote team members. The level looks enhanced over the time, we raised this issue with our managerial meetings, I think that is why it got enhanced. **[RQ1, RQ2, PRODUCTIVITY]**
- Schedule: Was high at the starting point, that time I remember code was freezed, and no new tasks were taken for couple of days. Schedule was restricted during the middle of the sprint, where each developer tries to meet the deadline. Boundaries of the sprint includes some code freeze for

release tasks, which allows some flexibility. I think this is acceptable in my team.[**RQ1, RQ2, PRODUCTIVITY**]

- Time Fragmentation: Was high, and I expect this as the typical case in the team, since every developer has his/her task to work on, however, some ad-hoc task could could be taken, and we can not refuse that, since such kind of tasks could important to the customer, and have to be done even if this delay some other tasks. [**RQ1, RQ2, PRODUCTIVITY**]
- Requirements Stability: Requirements are alway subjected to change in the team, I wonder how the value showing good level of requirements stability, for example, I remember we had to re-implement full feature as the received requirements were fully changed. It is also something out of hand, at least at present, as the project is relatively new, and looking for potential customers that need to be satisfied. [**RQ1, RQ2, PRODUCTIVITY**]
- Application Experience: As the team is still new, the experiences is still low in this domain. One month would not show big enhancement on the experience. I see no big difference on experience through the named period. [**RQ1, RQ2, PRODUCTIVITY**]

A.2.2.6 Answer on Q8:

I could easily read the data presented in the home page. Usually I used my mobile for daily use, and found it good

A.2.2.7 Answer on Q9:

Yes, somehow. I prefer to provide more options to the manager, which allowing the focus on one factor rather than the others. For example, in my team, I'm more interested to find out what is happening in the team with regards to communication rather than the requirements stability. But in different stages, I would prefer to focus on the requirements stability rather than anything else.

A.2.2.8 Answer on Q10:

I would suggest to have special view for each team, instead of having the option to view the dashboard of other teams, as well as adding notifications on some thresholds that the manager can determine

A.2.2.9 Answer on Q11:

3.5, after some actions taken, the communication were enhanced, that allowed the team to better resolve the dependency conflicts, and became better. **[RQ1, PRODUCTIVITY]**

A.2.2.10 Answer on Q12:

Yes, I'm satisfied taking in mind the relatively short period of running this application. **[RQ1, PRODUCTIVITY]**

A.2.2.11 Answer on Q13:

Very good, that would reduce overheads of closely following up with each team members, and allowing taking actions accordingly

A.2.2.12 Answer on Q14:

Most of them, according to my experience, I take in mind most of these factors, in addition to more factors. But was good to have them in such way

A.2.3 Answers by Manager3 (M3) for Team3 (T3)

A.2.3.1 Answer on Q1:

Yes, by end of each sprint. This helps me to judge the team performance, and the progress on their tasks. [**RQ1, PRODUCTIVITY**]

A.2.3.2 Answer on Q2:

At the end of each sprint, I review the completed tasks, and the expected tasks to be done, taking in mind the quality of the completed tasks, by tracking the bugs and issues. [**RQ1, PRODUCTIVITY**]

A.2.3.3 Answer on Q3:

3, although the team members are hardworking, but occasionally we face delays in delivering the task, in most cases, the delays are reasonable. [**RQ1, PRODUCTIVITY**]

A.2.3.4 Answer on Q4:

We faced high rate of turn over by the team members. Having new members in the team requires training and knowledge transfer, which causes some delay, and reduced the velocity of the team. [**RQ1, PRODUCTIVITY**]

A.2.3.5 Answer on Q5, Q6 and Q7:

- Team Cohesion: The relationships among the team members are good, that leads to good cohesion from the beginning. But due to the turn over in the team, the level of the cohesion looks vary, but in acceptable values. Actually, there is some improvement on the reading, I didn't acted according these values, as I believe in the good relationships among the team members. Also, I'm more interested in other factors. One more point, each of one in the team works on different area that limits the inter-collaboration within the team members here. Also no one is expected to work on others' tasks for the same reason. [**RQ1, RQ2, PRODUCTIVITY**]

- Communication: The communication is vital in my team, especially the communication with the team members in different locations. The communication is evolving by the time. But I did not act on this factor, since I'm satisfied with the level of communication on the team. [**RQ1, RQ2, PRODUCTIVITY**]

- Schedule: As we run Scrum, the schedule is usually tight, and we always try to complete the committed task per sprint. According to the readings, I tried to have less tasks for the next iteration, and that the reason of the good schedule value at the end of the study period.[**RQ1, RQ2, PRODUCTIVITY**]

- Time Fragmentation: Looks like the team was having low values for time fragmentation, investigated the issue, and found that there are some destructions on the main tasks by some side work for some team members. I asked these developers not to accept any ad-hoc requests before referring to me. And that why it got developed over the time. [**RQ1, RQ2, PRODUCTIVITY**]

- Requirements Stability: The project is still in its early stages, we expect requirements changes. However, we use to receive unclear requirements that need to be clarified. And this all taken care by the team members by proper communications with the product owner. **[RQ1, RQ2, PRODUCTIVITY]**

- Application Experience: Even though the project is new, the experience in the team developed rapidly, however, the high level of turnover makes the experience to be low so far. **[RQ1, RQ2, PRODUCTIVITY]**

A.2.3.6 Answer on Q8:

The application was easy to use, and I did not find any difficulties using it. But was alway looking for the page's link, it is something hard to reach. I suggest to have better URL for the ease of access.

A.2.3.7 Answer on Q9:

They were fairly enough, but I recommend to highlight the team members feedback with low values, to allow better investigations about the corresponding factor..

A.2.3.8 Answer on Q10:

To have better URL for the ease of access to the dashboard.

A.2.3.9 Answer on Q11:

If I said it was 3 in some question you asked me, I would say it became 3.5 after using the system. I found this during the sprint review later to the system using period. **[RQ1, PRODUCTIVITY]**

A.2.3.10 Answer on Q12:

Yes, it was enhanced.

A.2.3.11 Answer on Q13:

Overall talking, they are useful factors to explore in the team. Some looked important to me, others were not, as I declared previously. [**RQ1, PRODUCTIVITY**]

A.2.3.12 Answer on Q14:

They are kind of common-sense factors, that should be known by the manager about his/her team. But I don't say that the system is not important, on the contrary, I found it useful.

Appendix B

Team Members Instructions

Hi all,

You team was selected to be part of a case study as part of my dissertation of Master degree in Software Engineering, at Birzeit University.

You are kindly requested to join slack at our company work-space (https://exalt.slack.com). As a team member, each one of you has been added to slack channel for your team, where you will receive questions on daily basis, and you are encouraged to answer all of these questions. The study will take about 1 month.

Please consider the following points:

- You will receive six multiple-choice questions on each business day
- You may get the same question on different days
- You can only answer the questions of the current day
- You can change your answers during the day
- Your manager cannot see your answers; however, he can see your voting rate and ask you for better collaboration

- Your answers give your manager a comprehensive feedback on the impact of some factors on the team's productivity. The factors include communication, team cohesion, time interruptions and others

And finally, I appreciate your collaboration.

Appendix C

Managers Instructions

Before starting the study, I would like to list some points to be considered during that:

- The dashboard displays comprehensive information about the impact of some factors on the team's productivity. The factors include communication, team cohesion, time interruptions and others
- These information can assist you in better planning for enhancing the team's productivity
- The data is collected by asking direct questions to the team members
- We will use the first week's collected data as initial data of the team, so you are encouraged not to take any action based on the data you receive during the first week of the study
- interview will be done with you at the end of the study
- Team members will receive a questionnaire at the end of the study, will be aimed to evaluate the tool

Thanks for your collaboration in this study

Appendix D

Team Survey

Following Questions were used in the survey posted to the team members:

- Select your team
- I was easily answered the posted questions (Strongly disagree - Strongly Agree)
- Slack needed me long time to configure it on my computer, mobile, ... etc (Strongly disagree - Strongly Agree)
- I feel comfortable to receive questions on intervals (Strongly disagree - Strongly Agree)
- Notifications disturbed me during my work (Strongly disagree - Strongly Agree)
- I would prefer if I have all questions posted in one message (Strongly disagree - Strongly Agree)
- I found this application was good opportunity to keep in touch with my managers (Strongly disagree - Strongly Agree)

- Some of the questions I received expressed my current situation. For example; "I was working for late hours", or "I got distracted by other team members" (Strongly disagree - Strongly Agree)
- I feel this application helped resolving some issue was bothered me, and made me more productive. For example; "if you were blocked due to some communication gap, and eventually the communication became efficient" (Strongly disagree - Strongly Agree)

Bibliography

[1] Rajiv D Banker, Srikant M Datar, and Chris F Kemerer. “A model to evaluate variables impacting the productivity of software maintenance projects”. In: *Management Science* 37.1 (1991), pp. 1–18.

[2] Kent Beck et al. “Manifesto for agile software development”. In: (2001).

[3] Barry Boehm et al. “Cost estimation with COCOMO II”. In: *ed: Upper Saddle River, NJ: Prentice-Hall* (2000).

[4] Alistair Cockburn and Jim Highsmith. “Agile software development, the people factor”. In: *Computer* 34.11 (2001), pp. 131–133.

[5] David Cohen, Mikael Lindvall, and Patricia Costa. “An introduction to agile methods”. In: *Advances in computers* 62 (2004), pp. 1–66.

[6] Mauricio Cristal, Daniel Wildt, and Rafael Prikladnicki. “Usage of Scrum practices within a global company”. In: *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on*. IEEE. 2008, pp. 222–226.

[7] Tom DeMarco and Tim Lister. *Peopleware: productive projects and teams*. Addison-Wesley, 2013.

[8] Torgeir Dingsøyr et al. “Team performance in software development: research results versus agile principles”. In: *IEEE Software* 33.4 (2016), pp. 106–110.

[9] J Alberto Espinosa et al. “Familiarity, complexity, and team performance in geographically distributed software development”. In: *Organization science* 18.4 (2007), pp. 613–630.

[10] Hans-Christian Estler et al. “Agile vs. structured distributed software development: A case study”. In: *Empirical Software Engineering* 19.5 (2014), pp. 1197–1224.

[11] Martin Fowler and Jim Highsmith. “The agile manifesto”. In: *Software Development* 9.8 (2001), pp. 28–35.

[12] Fawad Ghafoor, Ibrar Ali Shah, and Nasir Rashid. “Issues in Adopting Agile Methodologies in Global and Local Software Development: A Systematic Literature Review Protocol with Preliminary Results”. In: *International Journal of Computer Applications* 160.7 (2017).

[13] R Green, T Mazzuchi, and S Sarkani. “Communication and quality in distributed agile development: an empirical case study”. In: *International Journal of Information Technology* 6.1 (2010).

[14] Dawson R Hancock and Bob Algozzine. *Doing case study research: A practical guide for beginning researchers*. Teachers College Press, 2016.

[15] Sean M Handley and WC Benton. “The influence of task-and location-specific complexity on the control and coordination costs in global outsourcing relationships”. In: *Journal of Operations Management* 31.3 (2013), pp. 109–128.

[16] James D Herbsleb and Deependra Moitra. “Global software development”. In: *IEEE software* 18.2 (2001), pp. 16–20.

[17] Helena Holmström et al. "Agile practices reduce distance in global software development". In: *Information systems management* 23.3 (2006), pp. 7–18.

[18] Emam Hossain, Muhammad Ali Babar, and Hye-young Paik. "Using scrum in global software development: a systematic literature review". In: *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*. Ieee. 2009, pp. 175–184.

[19] Nina Kamarina Kamaruddin, Noor Habibah Arshad, and Azlinah Mohamed. "Chaos issues on communication in agile global software development". In: *Business Engineering and Industrial Applications Colloquium (BEIAC), 2012 IEEE*. IEEE. 2012, pp. 394–398.

[20] Lucas Layman et al. "Essential communication practices for Extreme Programming in a global software development team". In: *Information and software technology* 48.9 (2006), pp. 781–794.

[21] Sarah Lewis. "Qualitative inquiry and research design: Choosing among five approaches". In: *Health promotion practice* 16.4 (2015), pp. 473–475.

[22] Likert and Rensis. "A technique for the measurement of attitudes". In: *Archives of Psychology* 22.140 (1932), p. 55.

[23] Mikael Lindvall et al. "Agile software development in large organizations". In: *Computer* 37.12 (2014), pp. 26–34.

[24] Katrina D Maxwell and Pekka Forselius. "Benchmarking software development productivity". In: *Ieee Software* 17.1 (2000), pp. 80–88.

[25] Claudia Melo et al. "Agile team perceptions of productivity factors". In: *Agile Conference (AGILE), 2011*. IEEE. 2011, pp. 57–66.

[26] Tahir Nawaz Minhas and Markus Fiedler. “Impact of disturbance locations on video quality of experience”. In: *EuroITV 2011 Workshop: Quality of Experience for Multimedia Content Sharing*. EuroITV 2011 Workshop: Quality of Experience for Multimedia Content Sharing. 2011.

[27] Parastoo Mohagheghi and Reidar Conradi. “Quality, productivity and economic benefits of software reuse: a review of industrial studies”. In: *Empirical Software Engineering* 12.5 (2007), pp. 471–516.

[28] Peter E Mudrack. “Defining group cohesiveness: A legacy of confusion?” In: *Small Group Behavior* 20.1 (1989), pp. 37–49.

[29] Anh Nguyen-Duc, Daniela S Cruzes, and Reidar Conradi. “The impact of global dispersion on coordination, team performance and software quality–A systematic literature review”. In: *Information and Software Technology* 57 (2015), pp. 277–294.

[30] Mahmood Niazi et al. “Toward successful project management in global software development”. In: *International Journal of Project Management* 34.8 (2016), pp. 1553–1567.

[31] Sandra L Ram, Hanna Oktaba, et al. “Productivity in Agile Software Development: A Systematic Mapping Study”. In: *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE. 2017, pp. 44–53.

[32] Balasubramaniam Ramesh et al. “Can distributed software development be agile?” In: *Communications of the ACM* 49.10 (2006), pp. 41–46.

[33] Sandra L Ramírez-Mora and Hanna Oktaba. “Team maturity in Agile Software Development: The impact on productivity”. In: *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2018, pp. 732–736.

[34] Kenneth S Rubin. *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.

[35] Per Runeson and Martin Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical software engineering* 14.2 (2009), p. 131.

[36] Syed Muhammad Ali Shah, Efi Papatheocharous, and Jaana Nyfjord. “Measuring productivity in agile software development process: a scoping study”. In: *Proceedings of the 2015 International Conference on Software and System Process*. ACM. 2015, pp. 102–106.

[37] Jeff Sutherland, Guido Schoonheim, and Mauritz Rijk. “Fully distributed scrum: Replicating local productivity and quality with offshore teams”. In: *System Sciences, 2009. HICSS’09. 42nd Hawaii International Conference on*. IEEE. 2009, pp. 1–8.

[38] Raoul Vallon et al. “Systematic Literature Review on Agile Practices in Global Software Development”. In: *Information and Software Technology* (2017).

[39] Stefan Wagner and Melanie Ruhe. “A Structured Review of Productivity Factors in Software Development”. In: (2008).

[40] Stefan Wagner and Melanie Ruhe. “A systematic review of productivity factors in software development”. In: *arXiv preprint arXiv:1801.06475* (2018).

[41] Laurie Williams and Alistair Cockburn. “Guest Editors’ Introduction: Agile Software Development: It’s about Feedback and Change”. In: *Computer* 36.6 (2003), pp. 39–43.

[42] Claes Wohlin et al. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[43] Qing Yang et al. “Identifying and managing coordination complexity in global product development project”. In: *International Journal of Project Management* 33.7 (2015), pp. 1464–1475.

[44] Robert K Yin. *Case study research and applications: Design and methods*. Sage publications, 2017.