RAC 2

The paper *"Scrum + Engineering Practices: Experiences of Three Microsoft Teams"*, published on IEEE Computer Society in 2011, presents the results of three Microsoft teams using SCRUM and nine engineering practices, which allowed them to improve their quality, productivity and estimation accuracy. The case study was first started by Laurie Williams from the North Carolina State University, who had contributions by Gabe Brown, Adam Meltzer and Nachiappan Nagappan who were part of the teams tracked at Microsoft. One issue that normally arises with SCRUM is a thing called Flaccid Scrum, which is a term coined by Martin Fowler, that refers to a team that uses only SCRUM for project managements and does not pay attention to the quality of the deliverable produced at each iteration. However, this can be prevented by creating a happy path for features, which is the easiest scenario to implement a feature. In addition, teams can pay attention to the quality of the product by implementing good engineering practices, like Microsoft which uses nine engineering practices with SCRUM. The nine practices are as follows:

1. Basic Scrum: Teams implemented the basic SCRUM methodologies and found that it allowed them to respond faster to changes in the requirements and therefore reduced risks. However, at the beginning the teams had a dip in productive while they got used to using the methodologies.

2. Planning Poker: It is a process to estimate the amount of hours it takes to implement a feature. The benefit is that it enables teams to know the amount of work it will take to complete a feature in one iteration, however this practice requires more upfront work at each session.

3. Continuous Integrations: This refers to constantly adding small changes to the main code and running tests. This is good since it increases the quality of the code, however it can generate merging issues and system problems.

4. Unit Test Driven Development: This is having unit tests for the code that are used to ensure the code does not show big errors, but it takes time to design the tests.

5. Quality Gates: These are criteria that allow developers know when a feature is complete. For instance, the feature is not considered complete if it does not meet all of the criteria set beforehand. This takes an extra expense of time as it requires more monitoring.

6. Source Control: It is a way to manage the code and its changes. Teams reported that it was useful to track changes, but the downside is that when the project is large managing these changes and ownership becomes hard.

7. Code Coverage: This means that each engineer makes sure that their code is covered by extensive test to eliminate dead code and areas that need better coding. However, it did increase the development time as test need to be created for several cases.

8. Peer Review: This is having peers review the code to eliminate faults that may have escaped the original developer.

9. Static analysis tools:Means using tools to catch errors and review the code. For instance, developers have to fix all the warning on a code before being allowed to continue coding. This reduced the amounts of defects in the code, but it also incremented the development time.

The authors based their conclusion on the results of the three Microsoft case studies. Although, this is not an experiment and therefore does not allow us to generalize for all the different teams that exist. It is evident that using SCRUM and engineering practices lead to improved productivity and product quality, since some team saw a 250% improvement on these areas. In short, the most important point from this paper and the findings of the research paper is that SCRUM methodologies alone do not guarantee the development of a good product. Therefore, to achieve good quality and increase productivity, SCRUM needs to be combined with other engineering practices that add value to the development team and product.