Scrum + Engineering Practices: Experiences of Three Microsoft Teams was published in 2011 by the International Symposium on Empirical Software Engineering and Measurement. The authors Gabe Brown, Adam Meltzer, and Nachiappan Nagappan are affiliated with Microsoft while the last author, Laurie Williams, is affiliated with North Carolina State University. In their paper, the authors start off by giving an explanation about the SCRUM process followed by a warning to what is called a Flaccid Scrum. Flaccid Scrum is a term created by Martin Fowler to represent teams that only rely on the Scrum project management practices instead of additional practices. With the mindset of having things simply "done", these teams would usually work with the focus of reaching the minimum quota. This would usually result in progress slowing down for these teams due to the fact that they had started paying less attention to the quality of their code, and instead paying more attention to making something work at the most basic situation. In order to avoid a Flaccid Scrum, a team would have to rely on additional engineering practices such as creating additional test cases, stress test the code, and improve the structure of one's logic instead of solely relying on the Scrum management practices.

Microsoft has also taken notice that simply relying on the Scrum management practices would result in a backlash, and instead enforced multiple engineering practices that would go along with the Scrum management practices. One of which is called continuous integration where a team would have to integrate their code into the main build system at least once a day. This will in turn create a constant focus on quality due to the fact that the cost is engineers having to deal with merge issues, and other issues pertaining to the inability to integrate branch codes into the main build.

Another engineering practice that Microsoft uses is called source control. Source control manages the changes of information such as pushing in a modified file or changing up the documentation of some code. This allows for managers to view what has changed, is the change necessary, and does the change have any bugs. Although source control allows other team members to view and manage changes to files, it can be difficult to manage as there can be cases where large amounts of files owned by another engineer had been modified.

The third engineering practice is called code coverage. Code coverage is a practice where engineers would perform unit tests in order to find dead code and areas that needed further testing. As a result of code coverage, engineers are given a rating of the quality of their code based on how well the unit tests performed, however, the unit tests only detected incorrect logic or computation, but not missing functionality.

The fourth practice that Microsoft uses is called peer review. Peer review is a practice where other engineers aside from the developer who wrote the code analyze the code. This would essentially improve the quality of the code as the reviewers would generally have input on how the code could have been better or detect a bug that the developer would have missed. At the expense of other developers' time, quality is more likely to be assured and bugs are more likely to be found.

The authors finally concluded that a combination of Scrum and engineering practices lead to improved productivity and product quality through comparing the defect density of a non-Scrum project from IBM vs a Scrum project from Microsoft. As shown in the graph, Scrum-A and Scrum-C had significantly less defects than that of IBM, which proves that a combination of Scrum practices and engineering practices will improve the overall quality and performance of a project. This truly astounded me as I assumed that just by having the Scrum process is enough, however, that is not the case. As shown in the graph, Scrum-B did not do well at all due to the fact that its testing efforts were really low. Only using Scrum without any other engineering practices will not be beneficial as the team would only develop the mindset of having things "finished" without thinking of doing tests for functionality, bugs, and ultimately going above and beyond for the code. The best combination would be to implement Scrum and other engineering practices.