CS115 HomeWork #2

For assignment 3, we were instructed to read the paper *Scrum+Engineering Practices: Experiences of Three Microsoft Teams*. This paper was published in the ESEM '11 Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement, by authors associated with North Carolina State University and the Microsoft corporation. This paper details the results of combining the agile practice of SCRUM with other, previously known, sound engineering practices by monitoring various teams in different departments at Microsoft.

One of the topics discussed early in the paper is the state of 'flaccid scrum' that teams of developers can fall into if they aren't careful. 'Flaccid scrum' means a scrum system that seems to do okay in the beginning but falls apart due to poor engineering practices and a lack of attention to detail. To quote the paper, "*Progress eventually slows for Flaccid Scrum teams, according to Fowler, because the team has not paid enough attention to the quality of the code produced during each iteration".* Although these teams may start off well and appear to be implementing features and accomplishing required tasks, the code generated might be sub-par and only tailored for the 'happy-case', the move favourable use-case in the developer's mind. This would of course imply that the test-engineers aren't doing their job correctly, otherwise mistakes would persist in the background long enough for a team to fall into a state of 'flaccid scrum'.

The Microsoft teams used a wide array of engineering practices to accomplish their goals. First off was playing 'planning-poker' when trying to come to a consensus on the difficulty of the sub-tasks assigned to a spring. Planning poker encourages a more genuine discussion on the difficulties faced by specific tasks (if done correctly) by forcing everyone to come up with their own prediction of a task's difficulty before being influenced by the feelings of others. It can also help to expose hidden assumptions that are shared by subsets of the group but not made explicit. Planning poker can backfire though if participants aren't careful to remain honest or fear deviating from the answers of their teammates too much. A technique related to testing that was used by the Microsoft teams is known as 'continuous integration'. Continuous integration gets it's name from the fact that it strives to continually, on as small of an interval as possible, run unit and integration tests to ensure that all code checked into the tree is correct and that new changes don't break compatibility to foreign modules. The beauty of continuous integration comes from designing systems around the concept that automatically handle the testing and analysis of code as it is checked into the group source tree. Another testing-related technique utilized by the Microsoft engineers is known as test-driven-development, or TDD. The defining characteristic of TDD is writing unit tests first (that fail), then writing code that can make those tests pass. This has the benefit of ensuring that tests are written early, rather than put off till the end and possibly forgotten. This has the effect of increasing the coverage of test-suites and causing bugs to be found earlier in development. The possible downside is having lazy engineers write bad tests that give the impression of a solid code-base, only for errors to show-up late in the project. Lastly, the Microsoft teams used the generals crum practices of short, daily stand-up meetings, maintaining the SCRUM project structure (project owner, scrum master, etc), and working in short sprints. The short sprints were found to be beneficial because it allowed the teams to react more quickly to changing business requirements (this was directly noted by a team that changed from 4 to 2 week sprints during the study).

The authors conclude that the combination of SCRUM with existing sound-engineering practices increases the productivity of the engineers on the team in question and statistically lead to better-working software being delivered closer to on-schedule than without the use of SCRUM techniques. They base this conclusion of the marked improvement in various categories, including teams having lower error rates when estimating work-loads and finding bugs earlier in the projects. They also showed that teams who utilize scrum are more likely to finish on time than teams that do not.

The part of this paper that spoke most to me was the reduction in estimation errors experienced by the teams at Microsoft. Being able to accurately estimate the time-to-delivery for a project is one of the most important parts of software-engineering, so any techniques that help a team do this is significant to me.