1.

In the article, Henderson mentions that the source code for Google is conglomerated into a massive repository, which anybody can edit. If an edit or an improvement passes the automated testing and is approved by one of the owners of the branch of the repository that the original code exists on, the code is updated. This does seem to create a small pitfall for the progress of code improvement, as branch owners will occasionally attest that their original code is superior and an edit or an improvement is not necessary.

I also found the concept that Google engineers are encouraged to freely spend 20% of their salary paid time on any project that they please to be an interesting proposition. On the one hand it provides a nurturing environment wherein creativity is encouraged, and innovation comes to the company naturally. On the other hand, it also has the potential to create a small itch in the minds of the engineers that might make it difficult for them to focus on their primary projects, which make up the remaining 80% of their time.

Hey Reinhold,  
While I agree that the simplest approach is often the most successful, I do not feel that Google’s current approach to software design could be described as simple. The implementation of the single repository and the reduction of the number of languages used streamlines the project, there is still an incredibly complex system of checks in place that ensure that the code which is committed is also correct and approved by owners of the original branch.

2.

In the past I have not had a direct usage of the agile development approach, but I did experience something like it during a development project I was a part of a couple of years ago. I enjoyed the process of moving through cycles of development and checking back with the project manager to check progress on requirements. This allowed the developers to make sure that their efforts were being used in accordance with the end goal of the project. It also made sure that each moving piece was produced properly and fit well with the others during integration. The downside of this experience was that as a developer, I didn’t get the chance to partake in the planning or anything else involved in the oversight of the project. In this project, my partners and I would be given an input format, an output format, and a time complexity constraint. It seemed more like homework and tedious labor, rather than being pert of a larger project. Agile on campus has proved itself to be difficult to accomplish due to the conflicting schedules of the members of our team. We plan to overcome this by meeting later in the evening once all of us are off of work and out of class.