Adapting Extreme Programming For A Core Software Engineering Course

The paper is about how extreme programming, an agile software development method can be adopted and used into a senior level software engineering class at North Carolina State University. We agree that agile isn't what it used to be and that it has turned into quick paced unpredictable environments using a lot of documentation. We haven't worked in an environment of what agile originally was, but we have worked in environments where there is a fast moving unpredictable environment that is more rigid, which is what the paper describes agile has transformed into. It makes sense that agile formed because industries were rapidly changing, and they couldn't stick to their very strict linear methodologies before because the market was becoming more unpredictable so they needed to be more lenient so they could change up depending on the market. It's interesting how software isn't the only industry that uses agile, but it makes sense that the software industry leaned towards using agile methodologies because requirements for technology are always changing. Technologies are always being adopted and built on so agile needs to be lenient so that depending on the requirements they're able to change their projects easily. It also focuses less on documentation than other methods previously used which slowed down the process so agile was very helpful because documentation wasn't a worry. We think it's cool that traditional methods have been taught alongside agile because depending on the industry, different methods are used so students being able to know and be able to adopt any method is very important rather than just knowing one and then joining an industry and going on it as they go. It's very important that in the North Carolina State University course, students tried different methods so that they could compare it to extreme programming so that they know what extreme programming actually looks like. Just hearing extreme programming makes us feel exhausted because it sounds very high intense and a stressful workload. It makes sense that a lot of students struggled doing extreme programming, especially the metaphor phase because it's quick moving and I also don't fully understand the purpose of it which makes sense why a bunch of people skipped it. We like the aspect that all the students have been in collective code ownership. Everyone gets to work on something, but it also makes sense that people felt like they lost ownership of parts of the code that they worked on. But pair programming also makes it less stressful because the burden isn't all on one person and it gets to be shared so you don't feel like you're rushing to meet deadlines by yourself. We do agree with the students about liking, not having to write documentation, it takes a large burden off and allows us to focus on our project, but we also do see the benefits of documentation, even though it is tedious. Not making documentation though does cause students to undervalued design, artifacts, design is very important, especially when it comes to human computer interaction and making a product that is easily accessible and usable by users rather than the developers. It made sense that students didn’t really refactor because the project duration was too short, they only had four weeks to be able to build what they needed to. Extreme programming sounds like it makes a lot of cuts on the typical agile methodology and each factor in agile is important for making a complete project. We think the benefits of extreme programming are it improves collaboration skills, which is very important for working in the industry, the students were able to get hands on skills with actually developing rather than putting a lot of focus and emphasis into documentation which would’ve taken time away from their development, continually integrating testing and building your project up is also very important because that’s how industries work because tech technology is constantly adapting and growing, we also thought the paired programming aspect was really interesting because it allows you to share knowledge between your partner and feel less of a burden of being solely responsible for your task, and most importantly the students were able to try different methodology, which allowed them to learn how to adapt two different workspaces as every industry or team has different practices that they use to build products. The problem we had was that four weeks definitely felt too short for the students to be able to really adopt their knowledge and practice, extreme programming, as they didn’t have much time for ref factoring, which is an important part of agile. They also weren’t able to get actual customers, and the professors had to be the customers which kind of hindered them on the use of the ability of their product as they won’t know how most actual users would use their product and figure out how they can make it easier for the users. Extreme programming seems to have a lot of benefits, but if we were to try it, it would have to be more structured and with a longer timeline that way we would be able to ref factor and have multiple cycles.