Seminar 2- Facebook

This seminar was given by Richard who is a network production engineer and Mike who works in Data centre management. This seminar focused on the operations n side of Facebook more so than a on the software development aspects. Facebook has roughly 2.1 billion users not counting the subsidiaries, WhatsApp with 1.5 billion, Messenger with 1.3 billion and Instagram at 800 million. In general, the data centres rely heavily on cooling and part of Mikes' job is to manage and maintain the servers. Failure is inevitable in software and operations however the most important thing is how you respond to the failure and n what you can learn from the failure these are key aspects to the operation of Facebook. If an individual finds an error in Facebook they are encouraged to flag it to the correct team and hand off the issue. Collaboration and communication and learning from mistakes appear to be a very high priority in the Facebook infrastructure.

Software Methodology:

The software methodology that is applied in Facebook is based from an agile framework and the tools and methods that are used are very flexible. Frequently, teams choose how they wish to develop the software and what tools they need. Facebook don't appear to be too concerned about how the coding is done and what methodology is utilised if productiveness and the code written benefits the system. In this methodology they will also ensure that they have some automated systems that will quantify and analyse the software developed for errors. In the event of errors, they have four teams the first will work together to keep the site running the second trees to reproduce the failure at a smaller scale the third will access the code for configuration changes and the fourth will dig through the data.

Use of Modelling:

Mike and Richard deal with the operations side of Facebook more than software development and as such each team decides on whether they use modelling. However, in their experience Facebook operates by form of effective communication not formal documentation. Their development process is much more cantered on Hackathons, social gatherings, and interaction within teams.

Testing:

It is important to note that testing won't always catch all the issues. They utilize unit test case which make up the bulk of tests by testing each component separately. This only tells us the component is error free it doesn't tell us if it integrates well into the current system. Integration tests are also utilized although there is less of these than the unit test cases. Integration tests are more complex as they test how well the components operate together. Another type of test that is implemented is Load testing. This is an important aspect in the testing process as it allows you to test the load on the system by simulating a peak environment. This gives an idea if concurrency is affected and the quality of the service being provided during high usage. They roll out changes in the system multiple times a day this ensures that the system is less likely to collapse and if it does the error can be found much quicker because less code must be analysed to find what caused the system to fail. In terms of testing to reduce the number of tests needing to be completed before rolling out they only test the methods that will be affected and not the whole system.

Software Quality & Refactoring

Facebook use to have a monitoring team however in recent years they have delegated the tasks of monitoring components to the teams involved. The teams are responsible for having alerts in place to make the developers aware of problems with the code. Facebook tries to automate a lot of these process as humans are generally unreliable. In this sense teams would have an incident manager which helps to steer the team the right direction to address code quality. They believe in getting thing done so code quality may not always be a high priority especially if they the time frame for deployment is quite short in comparison with the time thought needed to do the coding to perfection. This is where refactoring would happen, and this would be made a priority where features and updates would cease for a cycle to refactor the code. It is important to maintain the code as it allows others to join a team easier than if the codebase is ugly and hard to understand.