Software Development Processes

Lab Sheet 5 – Implementation

Introduction

This week in lectures, we looked at a number of unrelated techniques that can be used while implementing software. In this lab session, we're going to look at how some of these techniques could be applied to the scenario in the case study.

Task 1: Picking a Version Control System

The core benefits of a version control system: easier collaboration and the ability to rollback changes, are provided by all the commonly used tools. As long as the whole development team standardises on a particular tool, everyone will be able to make tracked changes to a single master version of the code. The decision on which tool to pick, therefore, relies on other considerations, such as:

- Where will the code repository actually be hosted?
- Will people be able to access the repository from where they are working?
- Is there a risk of an accidental leak if a third party is compromised?
- How much will it cost?

Research some of the version control systems discussed in the lecture, and review the costs for hosting the software components you identified in previous weeks, on a commercial basis. You might also want to look into the cost of a self-hosted solution if you were to build a server and use the company's existing data backup process. Discuss with the people around you what you think is the best option.

Regardless of the version control system you would use, certain pieces of information (e.g. login credentials, server addresses, encryption keys) should be prevented from being uploaded to the system. Then, if the version control system is indeed compromised, an attacker isn't then immediately able to gain access to other systems. This means that software components should be designed to load information from a configuration file that is available at runtime, but is stored and managed separately. Which components would need to store what information in this way? Discuss your thoughts with those around you.

Task 2: Using Pair Programming

Although pair programming isn't a perfect fit for every development team on every project, it is used by a significant number of companies very successfully. If you were to implement it in the scenario in the case study, who would you pair with whom? You will need to consider:

- Would the developers likely be working on the same part of the system?
- Do the developers have similar work patterns and schedules?
- Will they be able to physically sit at the same desk, or will they have to use remote tools?
- What is the relative experience level of the developers? How are they likely to interact?

Make notes of your proposed pairings. Why did you arrange them this way? Do you think that pair programming is a good idea in this particular scenario? Discuss your thoughts with the people around you.

Task 3: Class Discussion

Your lab tutor will gather everyone together with time before the end of the lab session to go through your selections and pairings. Did people generally arrive at similar conclusions? Did everyone just pick GitHub with some unconvincing reasoning? Were people's pairings radically different? Did their reasoning persuade you to adjust your pairings?