BSc (Hons) Computer Science

University of Portsmouth

Second Year

Software Engineering Theory and Practice

M30819

Semester 1&2

Hugh Baldwin

up2157117@myport.ac.uk

Contents

Lecture - Development Process Models

11:00 06/10/23 Claudia Iacob

System Requirements

- Functional Requirements
  - What does the system need to do?
  - What features does it need?
- Non-functional Requirements
  - Overall constraints of the system
  - What hardware does it need to run on?
  - How quickly does it need to run?
  - How fast should each endpoint of an API be?
- Requirements should be as specific as possible, to avoid confusion between teams
- They should also take into account what data is needed for the system to run, as well as it's computational requirements

Software Design

- More than just the design of the UI
- Also includes
  - Algorithms to be used for data processing
  - Use case modelling
  - Architectural design - How the application will be structured overall (frontend and backend and how they will be connected)
  - Database design - How will the data be stored in the database. What data needs to be stored?
  - Behaviour modeling - How will the parts of the system interact? How will errors be handled?

Software Implementation

- Coding and debugging
- Writing Documentation
- Continuous integration - deploy new versions of code automatically after testing functionality
- Version control - manage versions of code

Software Testing and Evaluation

- Evalutation - Are the non-fucntional requirements met?
- Validation - Is this the right system for the problem?
- Verification - Are the functional requirements met?
- Acceptance - Does the client agree that requirements are met, and will they accept and use it?

Software Development Lifecycle Models

- Iterative
- Incremental
- Aglie
- Reuse
- Waterfall

Iterative

- Iterate over the Specification, Design, Implementation and Testing
- The first iteration is usually a basic mockup to check that the client requirements are understood
- Gather feedback after each iteration, and keep improving until the final product is finished

Incremental

- Specify multiple increments based upon the specification
- Each increment could be as simple as one feature, or could itself need to be broken down more in a very complex system
- Each increment should be designed and implemented on their own, and then integrated and tested with all other increments, until the final product is reached

Agile

- Uses an incremental style
- Respond to change, make plans mostly in the short-term
  - Detailed plans for the week
  - Rough plans for the next months
  - Crude details for anything beyond that
- It’s better to collaborate with the customer, rather than set out requirements at the start and only communicating again once it’s finished
- The entire codebase is the responsibility of the team, this way if someone leaves the project, there isn’t a knowledge gap
- Pair Programming
  - One programmer writes code, the other reads the code and checks for logical errors
  - Swap between reading and writing often
  - The code is authored by both people
  - The pairs change very often - up to twice a day

Reuse

- Create the complete specification for the system
- Discover all available existing software and evaluate how they could be integrated
- Adapt existing software and write new systems to interface with it
- Integrate all reused and new software into a final system matching the specification

Waterfall

- Create the specification
- Create the complete design of the overall system
- Convert the design into code and integrate systems with each other
- Test the system as a whole
- Deploy and maintain the system. If there are any issues with the software, go back to the specification stage