Seng401 Lecture Notes Summary

Lecture Organisation

Lecture No.	Topic
1	Introduction
2	Evidence-based Software Engineer
3	Quality Assurance Standards
4	Validation and Verification
5	Validation and Verification
6	Validation and Verification
7	Non-functional Requirements and Software Performance Engineering
8	Source Code Quality Metrics and Refactoring
9	Technical Debts and Design Principles
10	Cognitive Biases and Software Engineering

Lec 1 Introduction

A **high-quality** software is

Internal Qualities	External Qualities
Well documented → user documentation and technical documentation)	Meets users' expectations → needs, functional requirements
Well-structured/decomposed/architected	Does not fail unexpectedly (NF)
No code duplication → reuse	Has no bug (NF)
	High performance, reactive (NF)

Software Development Life Cycle (SDLC) → process to **transform** users requirements into quality software products.

- Software quality depends on development process quality

Described as:

- Requirements specification → Design → Implementation → Testing/Verification & Validation → Deployment/ Delivery

Objectives:

- Reduce time-to-market → total time it takes to being a conception to market availability of product

- Improve product quality
- Reduce product cost

Improving quality = **↑ (immediate) cost & ↑ time-to-market**
Improving SDLC aims at reducing this conflict

Internal quality of design and source code guidelines:

- Avoid reinventing the wheel
  - OO design patterns
  - Map to existing algorithmic solution
  - Write sophisticated code → define problems precisely and look for existing algorithms
- Apply basic coding rules → Naming, Decompose, Reuse
- Document your code
  - Keep track of information/decisions/issues, as much as possible

---

Functional Requirements → Specified in terms of **WHAT** the software should do

- Defined by users/customers
- Defined in terms of stories; actions/reactions; input/ computation/output;
  - May include:
    - Preconditions → expected outcome **before** the requirement is fulfil
    - Postconditions → expected outcome **after** the requirement is fulfil

Non-functional requirements (or quality attributes) → additional constraints on **HOW** the software should fulfil functional requirements

- Examples: Portability, Security, Maintainability, Scalability, Usability, Reusability, Flexibility, Robustness etc…

Summary

Topic	**Notes**
SDLC	A process that transform user requirements into quality software products: - The process can be described as: Design → Implementation → Testing/Validation & Verification → Deployment & Delivery - Objectives are: Reduce time-to-market, Improve quality, Reduce cost - Aims at reducing the conflict of → improving quality = ↑ (immediate cost) & ↑ time-to-market
Internal quality of design and source code guidelines	- Avoid reinventing the wheel - Apply basic coding rules - Document code
Functional Requirements	Specify in terms of WHAT the software should do
Non-functional Requirements	Specify additional constraints on HOW the software should fulfil functional requirements