MidTerm Review

Don Chen

February 27, 2021

Exam on March 4th

1 Softare Quality

Lecture 4

- Quality and price are two things.
- User Satisfaction = The Important Qualities are High + Within Budget
- External qualities are visible to the user p7
- Internal qualities are visible to the developer
- Product qualities and Process qualities, p7
- Correctness: if it satisfies its requirements specification. p9
- Reliability: The software works most time, but fails in a small portion of time. p10
- Robustness: if it behaves reasonably even in unanticipated or exceptional situations. p11
- Performance(efficiency): related to external quality requirements for speed and storage. p15
- Usability: how easy a user can use the product. p16
- Verifiability: how easy it is to convince someone it is correct. p18
- Maintainability: how ease it is to modify after its initial release. p19

- Reusability: is reusable if it can be used to create a new product. p22
- Portability: if it can run in different environments. p23
- Understandability: how easy it is to understand requirements, design, implementation, and documentation. p24
- Interoperability, Productivity, Timeliness and Visibility: p30−33

2 Softare Principles

Lecture 5, 6

- Rigour: An argument is valid if the conclusion is logical,and it has mathematics support. Use between human.
- Formality: A language with a formal syntax and a precise semantics. Use between machines. Advantages: formality can look at all possible outcomes, and this can help to find anything is missed by written specs. Disadvantages: it takes a long time to learn it. The tools are not easy to use, and they are not complete. p12
- Separation of concerns: different concerns should be isolated and considered separately, reduce a complex problem to a set of simpler problems, parallelization of effort. p18
- Modularity: is a complex system that is divided into smaller parts called modules. It enables separation of concerns
  - Different parts of the system are considered separately
  - The parts of the system are considered separately from their composition
  - Good module design L6.P4
    - High cohesion: The components of the module are closely related. p24
    - Low coupling: The module does not strongly depend on other modules. low is good, but not zero coupling.
  - this allow modules be treated in two way:
    - As a set of interchangeable parts
    - As individuals
- Abstraction: is the process of focusing on what is important while ignoring what is irrelevant. Special case of separation of concerns. L6.P6

- Make an Abstract Data Type Abstract: don't say how it implemented, gave interface but not the implementation.
- Anticipation of change: is the principle that future change should be anticipated and planned for. Some technique to deal with change. L6.P10
  - Configuration management
  - Information hiding
  - Documentation
- Generality: is to use a more general solution to solve the problem. L6.P13
  - Good1: increase reusability.
  - Good2: general problem is easier to solve somtimes
  - Bad1: less efficient
  - Bad2: it is costly to have a general solution
  - Use more abstraction, usually help increase generality.
- Incrementality: solving problems by producing successively closer approximations to a solution. Get feedback, and improve it accordingly.

3 Module

Lecture 7

- Design for change: algorithm, safety standard. p5
- Product families: cars, phones
- Components of a module: a interface and a implementation(hide change in implementation). p13
- Information hiding: hidden from clients. Anticipating changes need to have information hiding. p16
- Example of Module p17
  - Record: only have data
  - Collection of related procedures(library): have behavior not state

  - Abstract object: only have one instance
  - Abstract data type: collection of abstract objects
  - Generic modules: abstract description for a family of objects, like generic module of stack of integers, strings, stacks.

4 Module Interface Specification

Lecture 9

- out and location function p7
- Why unit test: when merge module, we are confident problem occur when module interact with each other, not module itself.

5 Abstract Data Types

Lecture 10

- Read specification, semantics.

6 Functional Programming

Lecture 11, 12

- Functional Programming: Computation is treated a the evaluation of mathematical functions
  - No state, mutable data, only expression
  - No side effects
- List Comprehension L11 p14
- Map L12 p6
- Partial functions L12 p10
- Filter L12 p14
- Reduce p16

7 Modules with Ext. Interaction

Lecture 13

- Records
- Libraries
- Abstract Data types
- Abstract Objects
- Interfaces
- Generic Abstract Objects and Generic Abstract Data Types
- Inheritance
- External Interaction (Environment variable) p14

8 Generic MIS

Lecture 14

- Fill Semantics out:=? p6
- Abstract Objects and ADTs p18
- Abstract Objects use static method in Python
- Generic Modules − Stack exmaple p15
- State Variable for Generic Stack p18
- State Invariant for Generic Stack p19
- Semantics for Generic Stack p20

9 Object Oriented Design

Lecture 15

- Generic modules are implemented in Python via dynamic typing p7
- Inheritance p19
- UML Representation of Inheritance p25

10 Module Interface Des

Lecture 16

- UML Bank Account Example p4
- UML Interface p5
- Class Diagram and MIS: Both MIS and Diagram show state variable, inputs, methods, arguments MIS has semantics, UML doesn't, only syntax. Some UML just show there is an association, but it doesn't contain how it to be represented. MIS included it. p6
- UML Associations p9
- UML Aggregation p11
- UML Packages: IS_COMPONENT_OF p12
- Assumptions and Exceptions p13
- Exception p14
- Quality Criteria: Consistent, essential, general, minimal, high cohesion, low coupling, opaque.
- aovid exception p17
- essential p18
- setters and getters p19
- minimal p20

11 Module Decomposition

Lecture 17

- minimal p6
- increase essential p9
- increase minimal 11

- The USES Relation p17
- DAG − Directed Acyclic Graph p20
- use DAG in tree p22
- Hierarchy p23
- static relation: user relationship is static, so it doesn't dynamically change.

12 Module Guide

Lecture 18

- Reduce example p4
- List comprehension p6
- Upside down tree relation p8
- Association and DAG p9
- Module Decomposition p10
- IS_COMPONENT_OF p12
- Information hiding p14
- Three Top Conceptual Modules: hardware hiding, software decision hiding, behavior hiding p18
- Criteria for a good secret p21
- Rational Design Process view, Graphically, binary matrix p25
- Module Guide Template p28
- Traceability Matrices p29
- Verification p30