CS314 Exam 1 Review

James M. Bieman

Topics on Exam:
Notes 1, Text Chapter 1
Software product and process.
• What is software?
  - Information, not media.
  - Requirements, designs, code, ...
• Is it malleable?
• Software quality:
  - Quality attributes.
  - Different perspectives on software quality.
• Software classifications: COTS, custom, hybrid,
  stand-alone, embedded, real-time, network.

Notes 1, Text: Ch. 1 (cont)
• Historical trends: from expensive hardware
  and few users to cheap hardware and many
  users.
• The Software Problem:
  We need formal solutions to informally described
  problems.
• SW development myths: adding people to late
  projects, changing software is easy, ...

Notes 1, Text: Ch. 1,
and Ch. 2, Assignment 1
• Software processes:
  - Waterfall model: phases and limitations.
  - Evolutionary model.
  - Incremental development.
  - Agile development, for example
    - Extreme programming.
    - SCRUM
• A1: Using SW Development Tools – Git,
  GitHub, and Eclipse.
• Software disasters and successes.

Notes 2, Text Ch. 3-8 (pp. 67- 238)
Assignment 2
OO Design and Implementation Concepts.
• Review object-oriented concepts.
  - Objects:
    - Hidden: state representation and method
      implementation.
    - Public: behavior access through public
      interfaces.
  - UML class diagrams.

Notes 2, Text Ch. 2, 5, 11
Assignment 2
• Associations:
  - Non-hierarchical.
  - Composition/aggregation – whole-part.
  - Specialization/generalization (inheritance),
    polymorphism.
  - Use dependencies.
• A2: Refactoring a design.
  - Model-View Separation design pattern.
  - Façade design pattern.
• Relationship between designs and code.
  - Creating objects and their associations.
• Designing and implementing.
Notes 3, Text Ch. 11, A1, DS1, & A3: Verification & Validation

- V&V terminology:
  - verification, validation, testing, static analysis, formal verification, inspections, failure, fault, unit testing, beta testing, regression testing, ...
- Test scaffolding/harnesses: test drivers, stubs, oracles.
- JUnit
- Testing theory: why is testing hard.
  - Why is perfect testing impossible?
  - Other hard testing problems.
  - The notion of subdomains, equivalence classes, and boundary testing.

Verification & Validation

- Notes 3, Text Ch. 8 (pp. 240-269), A3.
- Black-box class or cluster testing.
  - Test based on ordering of class objects.
  - Test based on operations.
  - Test multiplicity.
  - Use a test oracle.
- Test plans: test names, strategy, description, verification.
- Test driver design: oracles, catching exceptions, reporting results.
- Use of fault models in testing.

Notes 3, Text (pp. 240-269), A1 (& A3): Verification & Validation

Fault & Failure RIP Model:

Conditions necessary for a failure to be observed
1. Reachability: Program location that contain the fault must be reached.
2. Infection: The state of the program must be incorrect.
3. Propagation: The infected state must propagate to cause some output of the program to be incorrect.

Notes 3, Text (pp. 258-279), A3: Verification & Validation

- White-box test coverage:
  - Statement/node coverage.
  - Branch coverage/edge coverage/decision coverage.
  - Condition coverage.
  - Definition/Use (DU) pair coverage, or the “all uses” criteria.
  - (Coverage tools: Emma)
- Coverage subsumption or “strength”.

Verification & Validation

- Tests and test paths:
  - Many tests can "cover" one test path.
- Inspections:
  - Performed on all kinds of software documents.
  - Focus on goals --- finding errors.
  - Participant roles.
  - Procedures.
- The need for simplicity in all software documents: code, designs, specifications, ...

Design Studios

- DS1: Code Inspection
- DS2: Agile Scrum User Stories
- DS3: Black Box Testing
Software Engineering Tools

- Eclipse
- Git, GitHub: version control, inspections.
- Junit: testing framework
- Emma: test coverage.