CMPT371 Final Review

1. Exam is shorter, reduced questions, exam is on computer
2. Questions:
   1. A lot of short answer questions
   2. True or False
   3. Blank Filling
   4. Few longer questions
   5. E.g. 3 reasons of peer view
3. Notable Topics
   1. Risk management
   2. QA
      1. Inspections and peer reviews
      2. Continuous integration
      3. Defect estimation
      4. Specifications
      5. Testing
         1. General principles
         2. Path-based testing
         3. Testing for object oriented system
         4. Test case design
         5. Tops
         6. Testing procures
         7. Testability
      6. Scheduling
      7. Teams and team dynamics
      8. Estimation
      9. Quality code
      10. Requirements elicitation
      11. Continuous integration
      12. Software entrepreneurship
4. Overarching Themes
   1. Proactive thinking, action to lower vulnerabilities
   2. Risk management
   3. Uncertainty, risk, flexibility
   4. Weakness at interface (technical and human)
   5. Opportunity costs (limited time, cost, …)
   6. Advantages of doing things early
   7. Pervasive impact of quality
      1. Why take longer time in concreated time
         1. Testing, doc, assertion …
   8. Considering the soft and hard
      1. People’s behaviour, motivations, user behaviours as a critical part of system
      2. Role of incentive
      3. Indirect effects (of pressure, long hours, etc.)
      4. Negotiation
5. Best keys practices
   1. Risk scanning:
      1. Look for risk come out (identified)
      2. …
   2. Accountable positions
      1. Risk officer
      2. Builder master
      3. …
   3. Peer view
      1. Inspection
      2. Code
      3. Documentation
      4. Clearly
   4. Risk-driven incremental delivery
      1. Requirement change
      2. Feedback from customer
      3. Find risk sooner
      4. If only one delivery, incremental requirements
      5. User wants sometime in the hand right the way
   5. Binary milestone
      1. What we done
      2. What not
   6. Daily build/continuous integration and smoke test
      1. Standard machine …
      2. Too many things need to group together
      3. Bug comes up
         1. System bug, refresh changes…
      4. Testability: 4 ways
         1. Separation concerns
            1. Each of these find the purpose
            2. Test isolation easily
            3. Test what that does do
         2. Specifications
            1. Gives the show what thing should be done
         3. Modularity
         4. Traceability
         5. Test Hook
         6. Assertion
            1. Allows you track
            2. Close to the fault
      5. How do you create test cases, how are you testing them?
   7. Regular status updates and brief meeting
   8. Defect estimation
      1. Defect setting
      2. Storage data
      3. Get indented group
   9. Rigorous risk-driven testing
      1. Boundary
      2. Assertion:
         1. Checking program logically and correctness
      3. With assertion, can’t use to assume
   10. Version control practice
       1. Git
   11. Time Estimation/boxing
       1. Highly
       2. Bias
       3. Tend to be narrow
   12. Process improvement and learning from mistake
       1. When things go wrong, we could be done …
   13. Change control mechanisms
   14. Throw-away prototypes
       1. A prototype a throw away thing
6. Risk Management
   1. Means of handling
      1. Acceptance
      2. Contingency planning
      3. Mitigation
      4. Avoidance
   2. Explicit, ongoing process
   3. Identification/Risk Exposure calculation/handling
      1. Catastrophe brainstorms
      2. Prioritization
   4. Ongoing risk scanning (monitoring)
      1. Previously anti
      2. New risk
   5. Help us …
   6. It is an activity
   7. Make sure …
7. Specifications
   1. High-level specification of what can be counted on
      1. Methods: pre/post/invariants
      2. Class as a whole: invariants, history properties
   2. Modularity benefits
      1. Users of abstraction
         1. Specifies guarantees on which can count
      2. Creators of abstraction
         1. Clear understanding of what can be free modified
   3. Specifications: Additional benefits
      1. Conceptual clarity
         1. Elimination of latent ambiguity
         2. Attention to exceptional/boundary condition …
8. Fakes, Mock, and Stubs
   1. Often during software development, we must test code a not against a full implementation of other code b, but against a stand-in …
      1. B may not be written
      2. B …
   2. Fakes
      1. Just return trivial
   3. Mocks
      1. Check context via assertions
      2. Help in spotting error
      3. Log
   4. Stubs
      1. Fake or mock code for particular method
9. Testing
   1. Structure and function of testing pipeline
   2. View of testing as more than finding bugs
      1. Slow or delay…
   3. Testing / schedule interactions
   4. Testing metrics (FFR)
   5. Test case design
      1. Test hooks
      2. Unit test
   6. Triage
   7. Test team link to development team
   8. Bug caught by testing vs. user
   9. Automated testing and code coverage
10. Testing 2
    1. Defect reporting processes
    2. Bug classifications
    3. Defect tracking system
    4. Risk-based testing
    5. Impact of separation of concerns on testing
    6. Relationships between tester and developers
    7. Test review
    8. Testing and peer review
    9. Who is doing testing
11. Testing 3
    1. Testability
       1. Test hooks
       2. Test harnesses
       3. Test stubs
12. Testing 4
    1. Regressions
    2. Alpha & Beta testing
    3. Testing escapes
    4. Triage (directed and otherwise)
    5. Testing focus
    6. Black box and glass box testing
    7. Test cases design
       1. Equivalence class
          1. Input filed
       2. Boundary value
       3. Latin square
       4. Orthogonal arrays
       5. Path coverage
          1. Node
          2. Transition
       6. Quality bug report
    8. Infeasibility of some test coverage
13. Coverage Procedure
    1. We identify the set of things we need to cover
    2. We develop a set of paths that include all of the things we need to cover
    3. We develop a set of concrete test cases that lead to these paths being covered
14. Deterministic Planning and Scheduling
    1. Scheduling
       1. Gantt chart
       2. CPM diagram
       3. CPM algorithm
       4. Critical path
       5. Float (total and free)
15. Estimation
    1. Target vs. estimate
    2. Key role of expectation setting
    3. Wide range of uncertainty
    4. Range vs. point estimates
       1. Range estimates key –
16. Peer views
    1. Peer reviews compared with testing: vs. and complementary
       1. Inspection
       2. Taking notes
       3. …
    2. Good and bad use of reviews
    3. Size and speed tradeoff
    4. Time and
17. Entrepreneurship
    1. Broad types of business model
       1. Market segment
       2. Pricing and licensing terms
    2. Financing mechanism, tradeoffs
    3. Importance of solid legal docs
    4. Marketing vs. sales
    5. Key role of first client
    6. Partner equity concerns
    7. Dispute resolution
18. Requirements
    1. Key importance of solid requirements
       1. Minimize work and chance of cancellation
    2. Structured and minimization of ambiguity
    3. Problem statement
    4. Political process: key stakeholder buy-in critical
    5. Sources of error
    6. Requirements leaks and change control
    7. Expectation setting