**Software Engineering Exam 1 Review**

Things to Study

1. Definitions
   1. Risk
   2. Validation
   3. Verification
2. Models
   1. Waterfall model
   2. V-model
   3. Prototype model
   4. Iterative/Incremental model
3. Do not need know
   1. Rational Unified Process (RUP)
   2. Adaptive Software Development (ASD)
   3. Any model that isn’t a UML
      1. No ERD, state machine, etc
4. Agile Software Engineering Process
   1. Know enough about fundamental principles behind agile
      1. Stuff in bold on agile principles
5. Extreme Programming
   1. Characteristics
   2. Issues
6. Requirements Analysis
   1. Requirements Capture Steps
      1. Elicitation, Analysis, Specification, Validation
   2. Types of Requirements
      1. Functional, Non-functional, Design constraints, Process constraints
7. Scrum
   1. Sprint
   2. What goes on during an iteration
   3. All the scrum stuff
8. Planning
   1. Deliverables
   2. Activities
   3. Milestones
   4. Critical Path
9. Crud/Scrud
10. Requirements Products
11. Use Case Diagrams
    1. Definitions
       1. Actor, etc
    2. Use Case Diagram for modeling behavior
    3. Use Case Detail
    4. Usage Scenario
12. UML Lecture
    1. Activity Diagrams to model flow
    2. Advanced AD Example (Flow is created by an external symbol) (time based behavior. Every certain period)
    3. Sequence Diagram
    4. Instance and Class Relationships (UML Diagrams)
13. Important Terms to define:
    1. Elicitation
    2. Validation
    3. Specification
    4. Requirements Management
    5. Risk
    6. Stakeholder

Extreme Programming

1. Communication
2. Simplicity
3. Feedback
4. Courage
5. Respect
6. Treat customer as team member
7. Issues
   1. All-or-nothing commitment to XP (partial is bad)
   2. Customer as team member makes requirements volatile
   3. Work products are limited
   4. Minimal design

Agile Manifesto

1. Individuals and interactions
2. Working software
3. Customer collaboration
4. Responding to change

Agile Principles

1. Working software is delivered frequently
2. Even late changes in requirements are welcomed
3. Face-to-face conversation is the best form of communication
4. Projects are built around motivated individuals, who should be trusted
5. Simplicity
6. Self-organizing teams (team itself, the process, and sprint schedule)
7. Regular adaptation to changing circumstances

Requirements Analysis

1. Elicitation
   1. Collecting user requirements
2. Analysis
   1. Understanding and modeling the desired behavior
3. Specification
   1. Documenting the behavior of proposed system
4. Validation
   1. Checking that specification matches requirements
5. In a Nutshell
   1. Identify user stakeholders
   2. Gather each user SH view of the system
   3. Analyze each SH statement for behavior, data objects, entities, and constraints
   4. Put analysis into a specs document and fix problems
   5. Check for correctness with user SHs (fix problems)
   6. If okay, take specs to Planning

Capturing Requirements

1. An activity in the software engineering process
2. This is the transformation of requirements into a specification
3. Clients rarely know precisely what they want

Types of Requirements

1. Functional
   1. Behavior, a transformation of data
   2. Processing of input into output
2. Non-functional
   1. Characteristic that software must possess
   2. Huge data, good response time, secure, high reliability
3. Design constraints
   1. Restricts design of system
   2. Target runtime platform, external entity interface, communication protocol
4. Process constraints
   1. Restricts the software engineering process
   2. Spire model to incorporate risk management stakeholders
   3. Agile methods for early release of some components

Requirements Products

1. Requirements definition
   1. A description of everything client wants the system to do and which entities are involved in each behavior
   2. Software must realize this definition
2. Specification
   1. Each requirement restated from developer perspective.
   2. Work product used by all other developer stakeholders
3. These two can be combined into a single document when referring to agile requirements products

Scrum

1. Agile method delivering highest business value first
2. Sprint length: 2 to 4 weeks
3. Sprint
   1. Scrum team selects subsets of work tasks for next iteration
4. Lots of meetings for task assignment, progress updates, problem resolution, brainstorming
   1. Planning, standups, reviews

Scrum Backlogs

1. Product Backlog
   1. Master list of things to do
   2. Items have 2 extra descriptors: business value and time to finish
2. Sprint Backlog
   1. A subset of product backlog items for that sprint
   2. Items selected based on
      1. Business value
      2. How long items will take
      3. How much team feels it can do that sprint
         1. Velocity – calculated from previous finished sprint items
   3. Items not finished return to Product Backlog
3. Product backlog can change during a sprint but the spring backlog **should not** be changed.

Scrum-specific Rules

1. Product Owner
2. User Story
   1. A free text, step-by-step description of a functional requirement from an end-user perspective
3. Scrum Master
   1. Enforcer of scrum rules

Use Case

1. A plain language format for specifying requirements
2. Something that the actor wants or needs to do with the software

Questions

* He could give us a software description that we would need to produce a model for. Be able to produce a use case diagram for a piece of software.

1. **What is a sprint?**

A subset of work tasks for a specific iteration in Scrum.

1. **What is a gantt chart used for?**

A gantt chart shows tasks assigned to resources over time and how much has been completed. It can also help identify which tasks are on the critical path and which tasks have slack.

1. **When would you use the waterfall model and why?**

It is acceptable to use the waterfall model when the client knows exactly what they want for fast software production.

1. **How does the waterfall process model differ from the iterative process model?**

The waterfall process model is for quicker development, but does not account for change like the iterative process model. Iterative has the same back structure as the waterfall model, but it also for a back track to one of the earlier steps in the process. This causes the iterative process model approach to take longer.

1. **What basic concepts do all agile methods have in common?**

All deal with the same basic actions (requirements, design, etc) and all accept change as part of the process.

1. **Why should the development of software follow an engineering process?**

Software usually involved stakeholders who depend on the working product. Production should be precise, thought out, and validated with the client (SH) before deployment. If software development did not follow an engineering process, it is at risk of not fulfilling the set of requirements/specifications and could fail overall.

1. **What are the five values of extreme programming?**

Communication, Simplicity, Feedback, Courage, and Respect

1. **Can you define the five values of extreme programming?**

**Communication** is the collaboration between stakeholders.

**Simplicity** is designing only for immediate needs and not for the future.

**Feedback** is the consideration of results from the stakeholders.

**Courage** is staying committed to Extreme Programming

**Respect** is the acknowledgement that every team member is valuable

1. **What is a stakeholder?**

A stakeholder is anyone who is affected by the software engineering process. A stakeholder can be a end-user, developer, investor, etc.

1. What normally takes place during a scrum sprint?
2. What is scrum?
3. **What are the capture steps in requirements analysis?**
   1. Elicitation – Gather requirements from the stakeholders
   2. Analysis – Understanding and modeling the desired behavior
   3. Specification – Documenting the behavior or proposed system
   4. Validation – Checking that specification matches requirements

Produces the software requirements specification (SRS)

1. **What are the fundamental principles behind agile?**
   1. **Working software** is delivered frequently
   2. Even late **changes** in requirements are welcomed
   3. Face-to-face conversation is the best form of **communication**
   4. Projects are built around motivated individuals, who should be **trusted**.
   5. **Simplicity**
   6. **Self-organizing teams** (team itself, the process, and sprint schedule)
   7. Regular **adaptation** to changing circumstances
2. **What are the stages of the software engineering process?**
   1. Requirements analysis and definition
   2. System Design
   3. Program Design
   4. Programming
   5. Unit/Integration/System Testing
   6. Delivery
   7. Maintenance
3. **What are the characteristics of the software process?**
   1. Prescribes the major activities and when each occurs
   2. Can produce intermediate product and final product
   3. Can be composition of mini-processes
   4. Each activity has entry and exit criteria (clear when it starts and stops)
   5. Sequential (know which activity is next)
   6. Each activity has goals
   7. **Constraints** can apply to one or more **activities**, **resources**, **products**
4. **Why is helpful to model a process?**

It provides a roadmap for development and documenting creates common understanding. It can also help find bugs in the process, even before development starts.

1. **What is one negative attribute of the V Model?**

Requirement changes are not detected until just before delivery where changes are very expensive.

1. **Placeholder**