Avery Peiffer

Total Points: 100 (Weighted points: 6%)

Q1. SWE Background & SDLC: [29.5 pts]

* How Software Engineering is different from simple programming? [5 pts]

Software is multidimensional and possesses several characteristics, all of which must be given attention for the software to reach its potential. Some of these characteristics involve no programming at all and are instead related to the practical application of the software. These characteristics include the software’s functionality, usability, efficiency, reliability, security, maintainability, scalability, and portability.

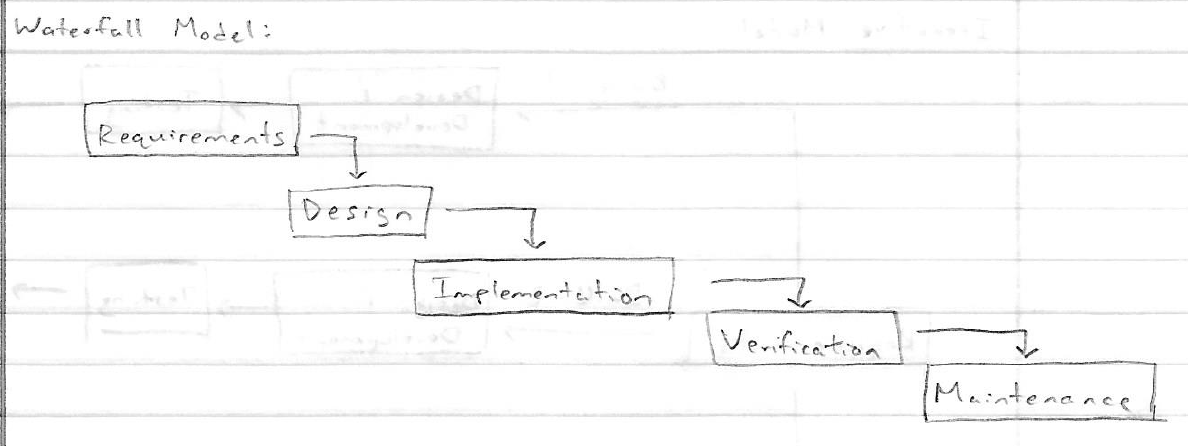
* Name & briefly define the five CMMI levels. [[2x5 = 10 pts]
* Initial: unpredictable process, poorly controlled, reactive
* Managed: process characterized for projects, reactive
* Defined: process characterized for organization, proactive
* Quantitatively Managed: process measured and controlled
* Optimizing: focus on continuous process improvement
* Name & state the purpose of three ISO standards used in the Software Engineering Industry. [2x3 = 6pts]
  + ISO/IEC 12207:2008: common framework for software life cycle processes with well-defined terminology to be referenced by industry; contains processes, activities, and tasks that are to be applied during the acquisition of a product or service
  + ISO/IEC 15504:2004: overall information on process assessment and its use in process improvement and process capability determination; describes how parts of the software suite fit together
  + ISO 9001:2015: outlines requirements for a quality management system when an organization needs to demonstrate its ability to provide products and services that meet requirements, and aims to enhance customer satisfaction through effective application of the system
* What is SDLC and define its various stages? [ 1 +5x1.5 = 8.5 pts. Approximately 1 page]

The software development life cycle (SDLC) is a well-defined and systematic approach for the development of a reliable, high-quality system, consisting of five stages. The first stage is requirements analysis, in which stakeholders hold thorough discussions to determine the purpose, features, and functionality of their intended software. These requirements are synthesized into a document to act as a guide for the following stages. Using this document, the organization enters the design stage, where the software and system architecture, design patterns, and static and dynamic models are designed. The hardware necessary for the system is also specified in this stage. A more comprehensive product requirement document (PRD) is generated using these design documents in combination with the requirement specification document.

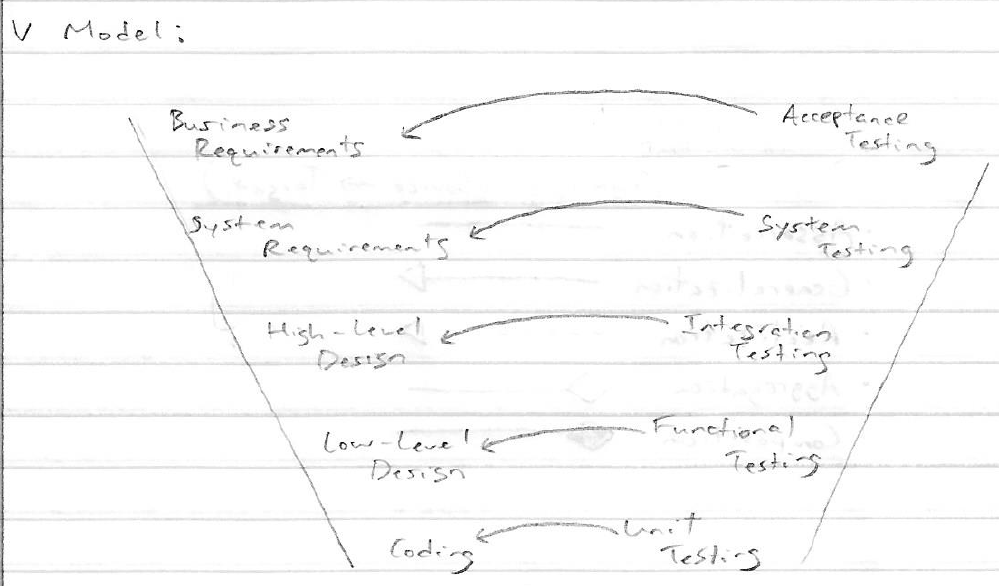
The longest and most important stage of the SDLC is the implementation stage, in which the actual coding is done. This code is written based on the design specifications outlined in the PRD. The system then moves into the testing phase, where the code written in the previous stage is tested to see if it meets all the requirements outlined in the requirements analysis stage. Types of testing that can be conducted include system testing, unit testing, acceptance testing, and integration testing. Finally, the system reaches the evolution stage, in which a finished product is delivered to the customer. The organization continues to monitor the system for problems that become evident only when external users begin using it. When these problems arise, the organization takes steps to solve them as necessary.

Q2. Software Development Methodologies: [25.5 pts]

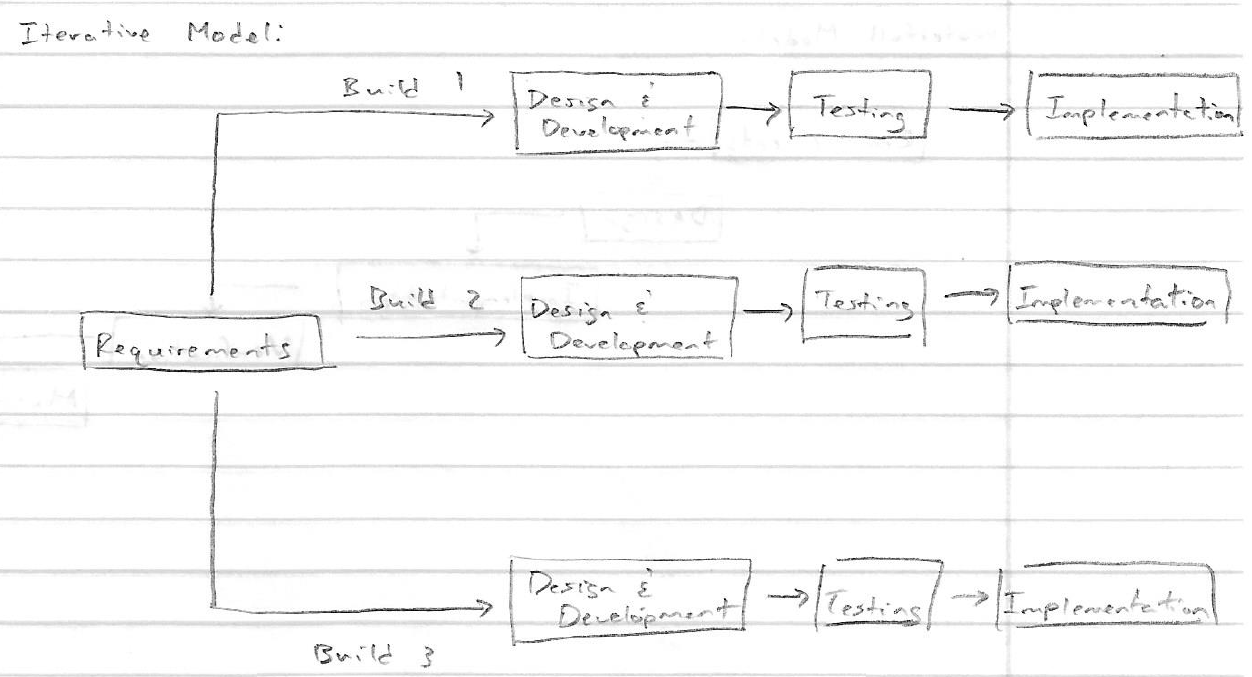
1. Name, briefly define, and draw diagrams for three popular software development methodologies. [3 x 1 pts + 3 x 2.5 pts + 3 x 2 pts = 16.5 pts. Approximately 2 pages]
2. Waterfall – a development process that occurs in a linear, sequential manner. A phase can begin only when all previous phases have been fully completed, and there is no overlap between any of the phases.



1. V-model (Verification and validation) – a sequential development process similar to the waterfall methodology, but with a form of testing associated with each phase of development. Like the waterfall methodology, a phase can only begin when previous phases are complete, and there is no overlap between phases.



1. Iterative – a model that does not begin with a full description of requirements. Rather, development is split into iterations of a fixed amount of time, during which a specific part of the software is defined, implemented, and reviewed. The early iterations contain simple implementations of a set of requirements which then evolve and become more complex with later iterations.



1. What are their key advantages & disadvantages of the SDLC methodologies in the previous question? [ 3 x 3 = 9 pts. Approximately 1/2 page]
   1. The waterfall methodology is useful for small projects that have static, well-defined requirements and a short duration. Additionally, the technology is well-understood and ample resources are available for the product’s creation. However, because there is no doubling back on any individual phase in the waterfall model, a project’s requirements must be clearly outlined at the very beginning of the development process. If a project has changing or ambiguous requirements, it becomes evident that the waterfall methodology is not an appropriate process to adopt for that project. This can be further compounded with especially long projects, which may have many hidden requirements that were not initially evident to an organization.
   2. The V-model process carries several of the same advantages as the waterfall methodology; it is most useful for projects with a minimal, understood scope and a short duration. However, the v-model is more useful than the waterfall methodology when the software’s quality is of primary importance. The addition of testing at each stage allows for an organization to root out and eliminate any quality issues much earlier in the development process than would be the case with the waterfall methodology. The V-model process also shares the downsides of the waterfall methodology, in that it is not useful for projects with ambiguous requirements or longer durations. It could be argued that the V-model is worse than the waterfall process for these longer, less clear projects, because additional resources are committed to testing the existing software despite the real possibility that requirements could change entirely.
   3. The iterative model is especially useful for a mid-sized or large organization that wishes to integrate its changing business needs into its software. The standardized, short iterations of one to four weeks allow an organization to receive near-continuous feedback on its product and make changes accordingly. This model is also useful for breaking up longer projects into simple parts, which allows an organization to delegate specific members to work in specific roles. While not explicitly covered, the relative complexity in the iterative model’s structure could be a downside if it is applied to the short, well-defined projects for which the waterfall model is best suited.

Q3. Agile: [26 pts]

1. What is Agile Methodology? [ 2 pts. Approximately 1 paragraph]

Agile is a software development philosophy built on the concepts of adaptability and iteration. The software development process is made up of iterations, each lasting 1 to 4 weeks, which allows the development process to respond to changing business needs. In doing so, the product can be developed in a turbulent, ever-changing environment. Practices that enable cost-effective change are promoted, and testing and continuous integration allow for rapid production with minimal up-front design. Agile processes often employ test-driven development, in which a harness of unit tests is written before the functional code, in order for developers to change the code aggressively.

1. Name and briefly define the three main Agile development flavors? [ 2x3=6pts. Approximately 1 small paragraph]

SCRUM is a methodology in which organizations are broken down into small units based on roles. Features of the product are broken down into user stories, and work is done in time-boxed iterations called sprints. Extreme Programming (XP) has a philosophy similar to that of SCRUM but is much more quality centered. XP programmers work in very short development cycles so that changes requested by the customer (who works on-site with the team) can be incorporated frequently. As such, XP requires highly skilled team members. Kanban is a methodology that has its roots in manufacturing. Jobs are broken down into small, discrete items which are written on a card and place on a board. Where SCRUM and XP limit the time allowed for an amount of work, Kanban limits the amount of work in any one condition.

1. Briefly define the various Sprint Meetings of SCRUM. [ 2x4=8 pts. Approximately 1 page]

* Sprint Planning – During the Sprint Planning meeting, the development team breaks user stories into tasks for the entire development team. The team sets a goal for the sprint and analyzes, evaluates, and selects doable tasks from the Product Backlog to meet this goal. The user stories are estimated by a metric chosen by the development team (story points), and they are chosen for the sprint based on the development team’s capacity.
* Daily Standup – The Daily Standup meeting is a brief meeting held every day where developers update each other on their previous progress and their plans for the upcoming day. This meeting streamlines communications and allows developers to synchronize their work. Developers usually discuss their work from the previous day, their plan for the current day, as well as any challenges or obstacles that have been presented. This allows the team to address any persistent challenges without the need for extra meetings or more complex forms of communication.
* Sprint Review – The Sprint Review meeting is where the team calculates the total number of story points completed for the sprint. This is used to calculate the velocity, which allows the team to measure the relative amount of work completed across multiple sprints. The velocity is then used to estimate the amount of work that will be done in the next sprint.
* Sprint Retrospective – The Sprint Retrospective meeting is similar to the Sprint Review meeting in that they occur at the end of the sprint, but it is much more focused on introspection and personal improvement than the Sprint Review meeting. The team discusses what went well during the sprint, what could be improved, and what the members will commit to improve during the next sprint. The Scrum Master leads this meeting and encourages the rest of the team to improve its practices for the next sprint to be better. This meeting provides a formal opportunity to focus on introspection and adaptation.

1. Name & briefly describe the various roles for the members in a SCRUM team. [ 2x3=6 pts. Approximately 1/2 page]

* Scrum Team – The Scrum Team is the team of developers that performs all work that is delivered to the customer. It is comprised of developers, testers, and designers who work together to deliver the requested and committed product increments. As with all teams, it is important for the Scrum Team to work well together by following the same goal and respecting one another.
* Scrum Master – The Scrum Master is responsible for helping the rest of the team understand Scrum theory, practices, rules, and values. This individual helps those outside of the Scrum Team to understand which of their interactions with the Scrum Team are helpful and which are not. The Scrum Master essentially ensures that the Scrum process functions as smoothly as possible by facilitating all in-person Scrum events, using various project management tools effectively, and making sure that the Scrum Team does not have a difficult time creating the product.
* Scrum Product Owner – The Scrum Product Owner is the individual who owns the product and is responsible for project evaluation, initiation, development, and marketing. They are the sole maintainer of the Product Backlog, and they ultimately hold power in making decisions about the nature of the product. They are also the individual that must approve any changes to be made to the product.

1. What are the product and sprint backlogs? [ 2x2=4 pts. Approximately 1 small paragraphs]

The product backlog is a collection of user stories, built during Sprint 0, that describe the work that needs to be done on the product. It is managed by the Scrum Product Owner and is the single source of truth for all changes made to the product. The Sprint backlog is the collection of user stories selected during the Sprint Planning meeting. It is built during the beginning of each sprint and is one week worth of work for the development team to accomplish.

Q4. Requirement Analysis: [9 pts]

1. Define Functional Requirements and Non -Functional Requirements? [ 2 x 2 = 4 pts. Approximately 1/4 to 1/2 page]

* Functional Requirements – The mandatory features and functionalities of the software that are directly experienced by end users. These are expressed in user stories and involve most of the work in building solutions that deliver value to the user. Input/Output, processing, and error handling are all types of functional requirements.
* Non-Functional Requirements – Non-functional requirements are system attributes, restrictions, constraints, performance, quality, etc. which are indirectly related to system behavior as experienced by the end users. Physical environment, interfaces, and security are all types of non-functional requirements.

1. Provide five examples for each kind in the question Q4a above for an ATM machine. [ 2 X 2.5 = 5 pts.]

* Functional Requirements – Ability to check balance, ability to withdraw funds, ability to deposit a check, ability to transfer funds, ability to deposit cash
* Non-functional Requirements – Total cost of installation, security against attacks, quickly return account information, system must be easy to use, ability to accept different types of credit cards­.

Q5. UML [10 pts]

1. Define the following UML relationships and draw the symbols: [5x2=10]
   1. Association – Two objects are related to one another
   2. Generalization – The source is a specialization of the generalized element and can replace it
   3. Realization – The original item is guaranteed to perform contract (used only for interfaces)
   4. Aggregation – The target is part of the source element, and the target can sustain without the source element
   5. Composition – The target is part of the source element, but the target cannot sustain without the source element

