**Final Exam – Software Engineering Survey**

1. Explain what is wrong with the notion that computer software does not need to evolve over time.

Answer (Section1.1):

Computer software must be revised as errors are discovered and corrected. Software must be updated to accommodate changes in the computing environment. Many times a customer will request changes to add new functions to an existing product or to accommodate changes in the business environment. Sometimes an older system will need to be reengineered to provide benefits to the user in a modern context. The bottom line is that software that does not evolve will eventually become unusable.

1. Describe the relationships among software engineering process, methods, and tools.

Answer (Section2.1):

Software process defines the framework that must be established for effective delivery of software engineering technology, by providing a context by which the software engineering methods are applied. Software engineering methods provide the technical how-to’s for building software. Software engineering tools provide automated or semi-automated support for software engineering process and methods.

1. Why it important for software processes to be agile?

Answer (Chapter 3 Overview):

Software process provides the stability, control, and organization to an activity to prevent it from becoming chaotic. Yet, modern software processes must only demand the activities, controls, and work products that are appropriate for the team and product to be produced – to ensure that it can accommodate changes easily and deliver a high quality software product.

1. Why are evolutionary models considered by many to be the best approach to software development in a modern context?

Answer (Section 4.1.3):

Because time lines for the development of modern software are getting shorter and shorter, customers are becoming more diverse (making the understanding of requirements even harder), and changes to requirements are becoming even more common (before delivery), we need a way to provide incremental or evolutionary delivery. The evolutionary process accommodates uncertainty better than most process

1. Describe the three key assumptions regarding software projects that every agile software process must address.

Answer (Section 5.3):

It is difficult to predict in advance which software requirements and customer priorities will change and which will not.

For many types of software design and construction must be interleaved, it is difficult to predict how much design is needed before construction can be used to prove the design.

Analysis, design, construction, and testing are not always predictable processes and this makes planning difficult.

1. List environment characteristics that can be considered toxic to software teams.

Answer (Section 6.3):

Frenzied work atmosphere

High frustration that causes friction among team members

Fragmented or poorly coordinated software process

Unclear definition of roles on the software team

Continuous and repeated exposure to failure

Differing and incompatible team member human traits

1. Describe the differences between software construction and software deployment.

Answer (Section 7.3.4 and Section 7.3.5):

Software construction is concerned with coding and testing of a software increment. Deployment is concerned with the delivery of an operation software product to the end-user, supporting the product during active use, and processing user feedback concerning the product usefulness.

1. What work products result from the requirements engineering process?.

Answer (Section 8.1):

The intent of requirements engineering is to provide stakeholders with a written understanding of the problem, the work products produced include usage scenarios, function and feature lists, and requirements models

1. List the types of models that might be used in requirements modeling and explain the role of each type of model

Answer (Section 9.1):

* Scenario-based (system from the user’s point of view)
* Data (shows how data are transformed inside the system)
* Class-oriented (defines objects, attributes, and relationships)
* Flow-oriented (shows how data are transformed inside the system)
* Behavioral (show the impact of events on the system states)

1. List the characteristics that should be considered when considering potential classes for inclusion in an analysis model

Answer (Section 10.1):

* Contains information that should be retained
* Provides needed services
* Contains multiple attributes
* Has common set of attributes that apply to all class instances
* Has common set of operations that apply to all object instances
* Represents external entity that produces or consumes information

1. Under what circumstances should requirements modeling be utilized for Web or mobile apps?

Answer (Section 11.5)

* Large or complex app to be built
* Large number of stakeholders
* Large number developers onapp team
* Development team members have not worked together before
* App success will have strong bearing on success of company

1. List the four design models required for a complete specification of a software design and the role of each.

Answer (Section 12.4):

Data design – high level model depicting user’s view of the data or information.

1. Describe the types of dependencies that can exist in an architectural design.

Answer (Section 13.7.2):

Sharing dependencies - represent the dependence relationships among consumers whose use the same source or producers who have the same consumers.

Flow dependencies - represent dependence relationships between producers and consumers of resources

Constrained dependencies - represent constraints on the relative flow of control among a set of activities

1. What are the steps used to complete the component-level design for a software development project?

Answer (Chapter 14.3):

* Identify all design classes that correspond to the problem domain.
* Identify all design classes that correspond to the infrastructure domain.
* Elaborate all design classes that are not acquired as reusable components.
* Identify persistent data sources (databases and files) and identify the classes required to manage them.
* Develop and elaborate behavioral representations for each class or component.
* Elaborate deployment diagrams to provide additional implementation detail.
* Factor every component-level diagram representation and consider alternatives.

1. Describe what is accomplished during the component qualification, adaptation, and composition activities of component-based development.

Answer (Section 14.7.2):

* Component qualification is the task of examining candidate library components and ensuring that they perform the function required for the new application.
* During component adaptation any component conflicts that surface when a library component is added to the new application are dealt with by wrapping the component (sometimes a new component must be engineered).
* During component composition, the qualified, adapted, and newly engineered components are used to populate the new application architecture.

1. What elements of a user interface design can be evaluated without building a working computer prototype?

Answer (Section 15.6):

1. The length and complexity of the interface specification (provides insight into learning effort required by user).

The number of user tasks specified and the number of user actions required to complete each (provide estimates of system efficiency). Number of tasks, actions, and states in the design model (imply the memory load imposed on the user).

Interface style, help facilities, and error handling protocol provide a general indication of complexity of the interface and its acceptance by the users.

1. Describe practices that enable designers to think about using patterns

Answer (Section 16.2.2):

1. Be sure you understand the big picture (context in which the software will reside)
2. Extract the patterns that are present at that level of abstraction in the big picture.
3. Begin your design with ‘big picture’ patterns that establish a context or skeleton for further design work.
4. “Work inward from the context” looking for patterns at lower levels of abstraction that contribute to the design solution.
5. Repeat steps 1 to 4 until the complete design is fleshed out.
6. Refine the design by adapting each pattern to the specifics of the software you’re trying to build.
7. Discuss how poor management decisions can impact software quality?

Answer (Section 19.3.6):

* Estimation decisions – irrational delivery date estimates cause teams to take short-cuts that can lead to reduced product quality
* Scheduling decisions – failing to pay attention to task dependencies when creating the project schedule may force the project team to test modules without their subcomponents and quality may suffer
* Risk-oriented decisions – reacting to each crisis as it arises rather than building in mechanisms to monitor risks and having established contingency plans may result in products having reduced quality

1. What is a formal technical review and why is one conducted? Outline the steps required to conduct a successful FTR?

Answer (Section 20.6):

The purpose of an FTR is to have a group of software engineers examine a discrete work product and determine whether on not the product is free of defects using the software specifications and standards as the review criteria.

To perform a successful FTR, the steps described in Section 15.6.3 are conducted.

1. Describe statistical quality assurance?

Answer (Section 21.6):

* Information about
* software defects is collected and categorized
* Each defect is traced back to its cause
* Using the Pareto principle (80% of the defects can be traced to 20% of the causes) isolate the "vital few" defect causes
* Move to correct the problems that caused the defects in the “vital few”

1. What are the key differences between validation testing goals and acceptance testing goals?

Answer (Section 22.7):

In validation testing, the test team seeks to ensure that each software function or performance characteristic conforms to its specification. In acceptance testing, the test team needs to ensure that the software works correctly for the intended user in his or her normal work environment.

1. Describe three control structure testing strategies.

Answer (Section 23.5):

Condition or branch testing -uses test cases that exercise every decision statement in the program.

Data flow testing - selects test paths (definition use chains) according to the locations of variable definitions and uses in the program

Loop testing -tests focus on the validity the repetition constructs (making sure that loops start and stop when they are supposed to)

1. Describe three partitioning strategies that can be used when performing class level testing for OO systems.

Answer (Section 24.5.2):

State-based partitioning - tests designed so that operations that cause state changes are tested separately from those that do not

Attribute-based partitioning - for each class attribute, operations are classified according to those that use the attribute, those that modify it, and those that do not use or modify the attribute

Category-based partitioning - operations are categorized according to the function performed: initialization, computation, query, or termination

1. How is software scope defined?

Answer (Section 31.3.1)

By defining how the software to be built fits into a larger systems, product, or business context and the constraints imposed by the context. Determining what visible objects the customer expects to see as output and what input objects are required to produce them. Determining the software function needed transform input to output and any special performance characteristics. Attempting to bound all information quantitatively, when possible, and descriptively, when numbers cannot be used.

1. Describe all activities that must occur in order to produce a Risk Mitigation, Monitoring, and Management Plan.

Answer (Chapter 35.7):

Risk Identification - determine the risks that are appropriate

Risk Projection - determine the likelihood that each risk will occur and the damage likely to occur

Risk Mitigation - figuring out strategies to avoid the risks

Risk Management and Contingency Planning - assuming each risk becomes a reality determine ways to limit their impact