

CS 350 Midterm Study Guide

Software Life Cycle Models

1. Three problems with the waterfall model are it requires capturing requirements early, it has no emphasis on active user feedback, and a working version of the program will not be available until late in the project time span.
2. One would use the waterfall life cycle model when requirements are fixed and less likely to change and for smaller software products.
3. Three different forms of prototypes are mockups, existing programs, and quick and dirty programs.
4. Circumstances when it is appropriate to use rapid prototyping model are when the customer does not identify detailed input, processing, or output requirements. It is also used when the developer may be unsure of the efficiency of an algorithm, the adaptability of an OS, or the form that human-machine interaction should take.
5. Some key practices of extreme programming are pair programming, writing of user stories, making of frequent small releases, simplicity, use of CRC cards, functionality is not added early, refactoring whenever possible, customer on-site presence, frequent integration of code, optimization done near the end, no overtime, etc.
6. Four phases of Rational Unified Process (RUP) are inception phase, elaboration phase, construction phase, and transition phase.
7. Inception phase deals with understanding requirements and defining scope. Major milestone is lifecycle objective (LCO). Elaboration phase focuses on requirements, design and implementation to produce executable architectural prototype, and software architecture description. Major milestone is lifecycle architecture (LCA). Construction phase focuses on design and implementation (first operational product or "beta" release). Major milestone is initial operational capability (IOC) and first external release ("beta"). Finally, transition phase makes sure the system has the right level of quality, bug fixes, trained users, adjusting features, adding missing elements, and delivering of the final product ("public release"). Major milestone is product release (PR).
8. Types of maintenance are corrective, perfective, adaptive, and preventive.

Software Requirements and Use Case Modeling

1. Subcomponents of the requirements development are elicitation, analysis, specification, and validation.
2. Subcomponents of the requirements management are change control, version control, requirements status tracking, and requirements tracing.
3. Some requirements elicitation/gathering practices/strategies are collaborative group workshops, interviews, surveys, questionnaires, ethnography (observing users at work), scenario analysis of user tasks (use case modeling), problem reports and enhancement requests.

4. Common risks/traps to avoid in requirements engineering are insufficient user involvement, creeping user requirements, vague and ambiguous requirements, un-prioritized requirements, gold plating, overlooked user classes, minimal specifications, and inaccurate planning.
5. Use case describes one way to use the system. It is centered on an actor's goal, describes steps to achieve that goal, is written in a series of interactions, and yields a result to the actor. In short, a use case describes a goal along with all of the things that can happen as a user tries to achieve that goal.
6. Four defining traits of an actor in use case modeling are it is a role (not necessarily a human), is external to the system, interacts with the system, and tries to achieve a goal.
7. Use case diagrams have relatively low information content. Actors are often shown as stick-people. Use cases are shown as ovals surrounding the name of the use case. Associations are drawn as connectors. Bidirectional associations have no arrowheads. Unidirectional associations have an arrowhead at one end. The direction of the arrow indicates who is initiating the interaction. Primary actors initialize interactions. Secondary actors have the interaction initiated upon them.
8. Items included in a typical use case description are (use case format) are name, unique ID, purpose, actors, preconditions, interactions/steps, extensions, post-conditions, etc.
9. Functional requirements are actions that software must be able to perform. They represent main product features/services and are expressed in terms of inputs and outputs. Nonfunctional requirements are overall qualities/attributes of the resulting system. They can be viewed as restrictions/constraints placed on a service. Failing to meet a nonfunctional requirement can make the whole system unusable.
10. Techniques used for validating requirements are requirement reviews, prototyping, designing black-box (functional) test cases, and defining user acceptance criteria/tests.

Software Project Planning and Tracking

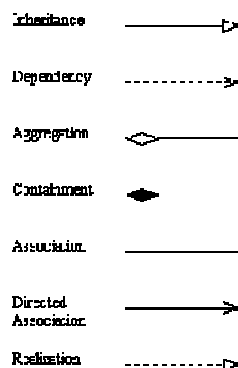
1. Critical path is the minimum time required to finish the project that is estimated by considering the longest path in the activity network.
2. Budget/cost estimation techniques are expert judgment, estimation by analogy, Parkinson's Law, pricing to win, schedule-driven budget estimation, and algorithmic budget estimation.
3. Three measures for estimating software size are lines of code (LOC), function points, and object/application points.
4. COCOMO 81 organic mode is relatively small, simple software project in which small teams with good application experience work to a set of less than rigid requirements. Semi-detached mode is an intermediate (size and complexity) software project in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements. Embedded mode is a software project that


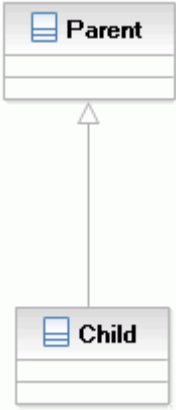
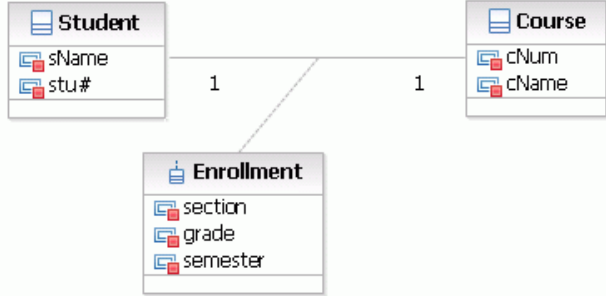



must be developed within a set of tight hardware, software, and operational constraints.

5. Know how effort is calculated in COCOMO 81 basic, intermediate, and advanced models.
6. Fifteen cost drivers are used in COCOMO 81 intermediate and advanced models.
7. Three models proposed in COCOMO II are application composition model, early design model, post-architecture model.
8. Seven cost drivers are used in the early design model of COCOMO II.
9. Seventeen cost drivers are used in the post-architecture model of COCOMO II.
10. The purpose of Earned Value Analysis (EVA) technique is to determine if the project is on track.
11. Know how to use COCOMO 81 and COCOMO II models to estimate effort, development time, and team size.
12. Know how to apply EVA technique to compute project progress indicators.

UML

1. Structure diagrams are used to model static/structural aspects of a software system. Structure diagrams depict the elements of a specification that are irrespective of time. Six structure diagrams are class diagram, object diagram, component diagram, deployment diagram, composite structure diagram, and package diagram.
2. Behavior diagrams are used to model dynamic/behavioral aspects of a software system. These diagrams depict behavioral features of a system or business process. Three behavior diagrams are activity diagrams, state machine diagrams, and use case diagrams.
3. Aggregation is a specialization of association, specifying a “whole-part” or “has-a” relationship between two objects. Aggregation is graphically rendered as a plain association with an open diamond at the whole end. It distinguishes the “whole” from the “part”. Composition is a form of aggregation with strong ownership. Semantics of strong ownership are parts are created after the composite itself and they live and die with it, an object may be a part of only one composite at a time, and the whole is responsible for creation and destruction of its parts. Composition is graphically rendered as a plain association with a filled diamond at the whole end.
4. Various UML Relationships:



Name	Relationship	Representation
Dependency	"using"	
Generalization / inheritance	"is a"	
Association	See aggregation and composition	
Aggregation	"has a"	
Composition	"has a" (strong)	
Realization	"realizes"	

5. Use cases are very detailed and typically define the actors, a brief description, pre-conditions, the main flow and any alternate flows, sub-flows and exception flows. It will also describe the state of the system at the end of each flow (post-condition). User scenarios are typically narratives versus the bulleted/numbered form of a use case. They incorporate individual user characteristics while outlining the tasks undertaken to achieve goals. Scenario is an elaboration of a use case into a precisely defined statement of system behavior. A scenario is represented as a use case plus a set of assumptions (initial conditions) and a set

of outcomes. Many scenarios may be derived from one use case and often a scenario can be a specific "path" through a use case.

6. Sequence diagram emphasizes the time ordering of messages. Two features that distinguish sequence diagrams from collaboration diagrams are object lifeline, or a vertical dashed line that represents the existence of an object over a period of time, and focus of control, or a tall, thin rectangle that shows the period of time during which an object is performing an action. Collaboration diagram emphasizes the organization of objects that participate in an interaction. Two features that distinguish collaboration diagrams from sequence diagrams are a path that shows how one object is linked to another by attaching a path stereotype to the far end of a link, and sequence number that indicates the time order of a message. Sequence and collaboration diagrams are semantically equivalent.