1. Describe the phases of the prototyping model for software development?

   Requirements are gathered by having the customer and developer meet and identify whatever objectives and requirements they can. Quick design follows, focusing on representation of the software that will be visible to the customer. A prototype is constructed by the developer and evaluated by the customer and used to refine the requirements. Iteration occurs and the prototype is tuned to satisfy the customer's needs.

2. What are the primary advantages of the component-based process model for software engineering?

   Component-based process models promote software reuse and reusability and can result in: 70% reduction in development cycle times, 84% reduction in project costs, and 70% increase in productivity.

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

   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 models, allows the delivery of partial solutions in an orderly and planned manner, and most importantly, reflects what really happens when complex systems are built

Describe the role of customers and end-users on an agile process team?

Customers and end-users participate as full collaborators on agile process teams. They are the source of information used to create use cases and provided needed information on the business value of proposed software feature and functionality. They also provide much needed feedback on operational prototypes during incremental delivery of software increments.

.

List the key issues stressed by an agile philosophy of software engineering.

The importance of self-organizing teams

Communication and collaboration between team members and customers
Recognition that change represents opportunity
Emphasis on rapid delivery of software that satisfies the customer

A common problem during **communication** occurs when you encounter two stakeholders who have conflicting ideas about what the software should be. That is, you have mutually conflicting requirements. Develop a process pattern (this would be a stage pattern) that addresses this problem and suggest an effective approach to it.

Pattern Name. *Conflicting Stakeholder Requirements*

Intent. This pattern describes an approach for resolving conflicts between stakeholders during the communication framework activity.

Type. Stage pattern

Initial context. (1) Stakeholders have been identified; (2) Stakeholders and software engineers have established a collaborative communication; (3) overriding software problem to be solved by the software teams has been established; (4) initial understanding of project scope, basic business requirements and project constraints has been developed.

Problem. Stakeholders request mutually conflicting features for the software product under development.

Solution. All stakeholders asked to prioritize all known system requirements, with resolution being to keep the stakeholder requirements with highest priorities and/or the most votes.

Resulting Context. A prioritized list of requirements approved by the stakeholders is established to guide the software team in the creation of an initial product prototype.

Related Patterns. Collaborative-guideline definition, Scope-isolation, Requirements gathering, Constraint Description, Requirements unclear

Known Uses/Examples. Communication is mandatory throughout the software project.

Provide three examples of software projects that would be amenable to the waterfall model. Be specific.

Provide three examples of software projects that would be amenable to the prototyping model. Be specific.

Software applications that are relatively easy to prototype almost always involve human-machine interaction and/or heavy computer graphics. Other applications that are sometimes amenable to prototyping are certain classes of mathematical algorithms, subset of command driven systems and other applications where results can be easily examined without real-time interaction. Applications that are difficult to prototype include control and process control functions, many classes of real-time applications and embedded software.

What process adaptations are required if the prototype will evolve into a deliverable

If a prototype is evolved into a delivery system or product, it begins with communication. The software engineer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. The prototype serves as a mechanism for identifying software requirements. If a working prototype is built, the developer attempts to make use of existing program fragments or applies tools that enable working programs to be generated quickly.

Provide three examples of software projects that would be amenable to the incremental model. Be specific.

Each linear sequence produces deliverable “increments” of the software for example, word-processing software developed using the incremental paradigm might deliver basic file management, editing and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment, and advanced page layout capability in the fourth increment. The process flow for any increment may incorporate the prototyping paradigm. Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.

As you move outward along the spiral process flow, what can you say about the software
that is being developed or maintained?

As work moves outward on the spiral, the product moves toward a more complete state and the level of abstraction at which work is performed is reduced (i.e., implementation specific work accelerates as we move further from the origin).

Is it possible to combine process models? If so, provide an example.

The process models can be combined, each model suggests a somewhat different process flow, but all perform the same set of generic framework activities: communication, planning, modeling, construction, and delivery/feedback. For example the linear sequential model can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner. In cases, where the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human-machine interaction should take. In these, and many other situations, a prototyping model may offer the best approach. In other cases, an incremental approach may make sense and the flow of Spiral model may be efficient. Special process models take on many of the characteristics of one or more of the tradition

What are the advantages and disadvantages of developing software in which quality is "good enough"? That is, what happens when we emphasize development speed over product quality?

The advantages of developing software in which quality is ”good enough” is that the product or software will meet the deadline, it may however lead to the delivery of software that is low in quality and requires time to improper the quality. When speed is emphasized over the product quality it may lead to many flaws, the software may require more testing, design and implementation work then done. Requirements may be poorly defined and may need to continuously change. Half hearted and speed may cause the risk management to fail to detect major project risks .Too little quality may result in quality problems and later rework

Are the Unified Process and UML the same thing? Explain your answer

UML provides the necessary technology to support object-oriented software engineering practice, but it does not provide the process framework to guide project teams in their application of the technology. Over the next few years, Jacobson, Rumbaugh and Booch developed the Unified Process, a framework for object-oriented software engineering using UML. Today, the Unified Process and UML are widely used on OO projects of all kinds