**Department of Computer Applications**

**23mx21 – Software Engineering**

Answer the following questions to validate the learning outcomes from “Software

Process Models” presentation

**Qsn1.** What problems might a software development team face if it does follow any life cycle models during development of a large software product?

**Ans 1**. Strictly following a life cycle model in practice may not always be feasible and in practice there may be some modifications and flexibility in adopting any model. However, every model will have some negative aspects if followed strictly which can be broadly as follows: -

(a) Rigidity: In the life cycle models that have a sequential approach like the Waterfall model, where one phase needs to be completed before the next stage can begin, the rigidity leaves less room for modification or flexibility to adopt to unexpected requirements or challenges during the project development.

(b) Uncertainty: While models like Incremental model, RAD, and Spiral model do provide a means for continuous interaction with customers and seek customers feedback during the project development stage , such approach does have uncertainties too in terms of the tentativeness of the final requirement and expectations, which can affect the proper planning and execution of the product development

(c) Models that have room for continuous feedback and interaction with customers at all stages of development run the risk of deviating from the original goals and requirements, thereby having a larger impact on planning and budget estimates.

(d) Clear and un-ambiguous communication is crucial in any development project, but this can be challenging to maintain among team members, and other project stakeholders, especially in larger teams or distributed environments and in models such as Evolutionary model, where the parameters and requirement are not clearly defined and are likely to be evolving as the project develops..

(e) Resource constraints: Life cycle models may assume the availability of certain resources, such as skilled team members or specific tools and technologies. However, resource constraints, such as budget limitations or staffing issues, can hinder the team's ability to execute the project effectively.

(f) In models that follow strict deadlines, quantifiable deliverables like the Waterfall model, can have an adverse effect on the project implementation to achieve targets in specified time frame which can result in compromise in quality of product, inadequate testing and increase risk of post deployment issues arising.

**Qsn 2.** Which phase consumes the maximum effort for developing a software product?

**Ans 2.** The phase that consumes the maximum effort in developing a software may vary depending on the specific project, its requirements, and the methodologies employed. However, it would be good practice to adopt the strategy of 60 : 15 : 25.

60 percent of effort should involve planning the project, which should include also the communication back and forth with the client to understand clearly the clients’ requirements, the final objective, etc. The design and analysis, modelling of the project should form a major part of this 60 percent component.

15 percent of effort should be the coding and actual software creation and development part. In the era of modern sophisticated, user-friendly software development and coding tools available, coding should be somewhat of an easy part and require less time and effort if there has been adequate deliberation, brainstorming and efforts put in the earlier phase.

25 percent of effort should be devoted to the testing phase of the product so that the product meets the expected benchmarks. Preparing comprehensive test cases ensures thorough and rigorous testing before the product is deployed. This calls for adequate attention and thus may take a larger part than the coding phase. This phase can also include in-situ testing after deployment at the client premise.

**Qsn 3.** Write an example of a software project for which the waterfall is not suitable.

**Ans 3**. The Waterfall model may not be best suited for Startup type of business ventures.

1. In a Startup venture, the business itself is new, the clients as well as the developers do not have adequate domain knowledge of the field they are entering, the market requirements and viability are not known and so a pre-fixed life cycle model like Waterfall is not suitable.
2. The Startups require a lot of flexibility, and constant interaction between the customer and developers, which the Waterfall model doesn’t provide for.
3. An example of a startup venture could be – developing a mobile application to provide an entirely new idea that defines a new way of communicating with our banking system and carrying out transactions. This new idea by the startup may not have existed before, but the customers intend to venture into it and explore the possibilities. So here the different phases of the Waterfall model cannot be adequately and clearly stated or defined.

**Qsn 4**. What do you mean by the “99% complete” syndrome in software development?

**Ans 4**. The "99% complete syndrome" refers to a common occurrence in software development where a project seems to be nearly finished, but the last 1% of work takes disproportionately longer than expected to complete. This phenomenon can happen for various reasons:

(a) Polishing and Bug Fixing: The final stages of a software project often involve polishing the user interface, fixing bugs, and addressing edge cases. These tasks can be more complex and time-consuming than initially anticipated.

(b) Integration Challenges: Integrating different components or systems can uncover unforeseen compatibility issues or require additional adjustments to ensure smooth operation.

(c) Testing: Comprehensive testing is crucial to ensure the quality and reliability of the software. As the project nears completion, more thorough testing often reveals unexpected issues that need to be addressed.

(d) Scope Creep: Additional features or changes may be requested late in the development process, extending the timeline for completion.

(e) Optimization:Developers may find opportunities to optimize the code or improve performance during the final stages, which can require significant time and effort.

(f) Deployment and Configuration: Setting up the software for deployment and configuring it to work in different environments can present unexpected challenges that prolong the final phase of development.

The 99% complete syndrome highlights the importance of thorough planning, realistic scheduling, and effective communication throughout the software development process to minimize unexpected delays in the final stages.

**Qsn 5**. If a developer has already experienced in developing payroll software for different customers; for developing payroll software for another customer, which life cycle model is recommended? Justify the reason.

**Ans 5**. The Incremental model would be recommended for this scenario.

(a) The developer already has a good idea of the different stages of the product development and as to how the project will unfold.

(b) The developer is in a position to quickly deliver a basic functionality model that the customer can use in a limited capacity while the other modules are being developed incrementally.

(c) The developer can give a sense of confidence to the customer about their credibility in the field which helps develop a long-term strong relationship.

(d) There is adequate room for flexibility to tweak the already implemented product to the specific requirements of the customer as the experience is already there with the developer.

(e) Since there is no specific time frame indicated by the customer, the incremental model gives the developer the opportunity to develop different stages as per schedule that can best produce the final product.

(f) Even from the business and financial perspective, the developer has a good opportunity to maximize ROI, having adequate experience in the domain, and can easily convince the customer to pay high remuneration for a very sound and tested product.

**Qsn 6**. Suggest a suitable life cycle model for a customer who is unsure of his/her requirements and is likely to change requirements very frequently. Write the reasoning behind your answer.

**Ans 6.** It is recommended to go for the Evolutionary model of software development life cycle.

1. Since the customer is new to the business – the customer is likely to be open to new ideas and suggestions that will contribute to the final product and customer satisfaction. In the Evolutionary model, there is continuous engagement with the customer at all stages of development.
2. The customer can be guided and shown the different software models in the proposed business domain, which will help the customer formulate their requirements.
3. Since the customer is not very sure of the requirements, the developers have an opportunity to come up with some prototype models that will help formulate some concrete ideas for both the customer and developer.
4. The Evolutionary model also has the possibility of the customer finding the proto-type model itself to their satisfaction there by the development of the product can be expediated with full quality improvement. There is the luxury of switching between the Throw away model approach and the Evolutionary prototype approach.
5. The model has a very sound, effective and immediate means of getting feedback and customer satisfaction reports. It provides for iterations and revisiting the product from different perspectives each time, thereby incorporating necessary changes and new ideas at every stage is easy to implement.
6. This model also has the flexibility to accommodate the customers’ new and evolving requirements there by earning the trust and goodwill of the customer.
7. The Evolutionary model gives more opportunities than other models to gain the confidence of the customer by being in the journey all along from the start to completion, there by having a long term satisfied customer relationship.

**Qsn 7**. For a project beset with many risks, would you recommend the prototyping or the spiral model? Justify the answer.

**Ans 7.** For a project beset with many risks, the Spiral model would be a more suitable choice compared to prototyping. Some of the reasons can be as follows:

(a) Risk Management: The Spiral model is specifically designed to address and mitigate project risks effectively. It incorporates risk analysis and risk management activities into each phase of the development process. By identifying and addressing risks early and iteratively, the Spiral model helps minimize the likelihood of costly surprises or setbacks later in the project lifecycle.

(b) Iterative Approach: Like prototyping, the Spiral model follows an iterative approach to development. However, unlike prototyping, which typically focuses on building a simplified version of the final product to gather feedback, the Spiral model emphasizes iterative refinement and risk mitigation. Each iteration in the Spiral model involves a cycle of planning, risk analysis, development, and evaluation, allowing the development team to address risks and uncertainties incrementally while gradually building and refining the software product.

(c) Flexibility: The Spiral model offers greater flexibility compared to traditional waterfall-based models. It allows for concurrent development activities and accommodates changes and adjustments throughout the development process. This flexibility is particularly valuable for projects with high levels of uncertainty or rapidly evolving requirements, as it enables the development team to adapt and respond to changing circumstances and risks effectively.

(d) Emphasis on Verification and Validation: The Spiral model places a strong emphasis on verification and validation activities, including testing, reviews, and demonstrations, at each iteration. This ensures that the software meets quality standards and performs as expected, reducing the likelihood of defects and mitigating project risks associated with poor quality or inadequate testing.

(e) Incremental Delivery: While the Spiral model does not necessarily prioritize early and frequent delivery of working software increments, it does support incremental delivery through its iterative approach. By delivering incremental releases of the software at the end of each iteration, the Spiral model allows stakeholders to validate progress, provide feedback, and make course corrections as needed, reducing project risk and increasing stakeholder confidence.

Overall, the Spiral model's emphasis on risk management, iterative development, flexibility, and verification/validation makes it a more suitable choice for projects that are beset with many risks compared to prototyping. By systematically addressing risks and uncertainties throughout the development process, the Spiral model helps ensure that the project stays on track and delivers a high-quality software product that meets stakeholders' expectations.