Teste 01
Victor Olimpio

2.5. Umbrella activities occur throughout the software process. Do you think they are applied evenly across the process, or are some concentrated in one or more framework activities?

Sim, as atividades de processo de software ajudam a equipe de desenvolvimento a controlar o progresso, a qualidade, mudanças e risco do sistema em desenvolvimento. As " Umbrella activities", por sua vez, possuem em seu escopo tópicos que auxiliam essa abordagem que pode ser implementada durante todo o projeto como :
- Análise de risco
- Garantia de Qualidade do Software
- Revisões técnicas
- Medição
- Gerenciamento de Reutilização

Retirado de : Seção 2.2.2 - Umbrella Activities - Software Engineering A Practitioner's Approach eighth edition - Página 17.

.

4.10. It is possible to prove that a software component and even an entire program is correct. So why doesn't everyone do this?

Através do processo, utilizando ferramentas da engenharia é possível perceber o caminho que o sistema irá percorrer. Uma definição bem estruturada do problema, assim com um escopo bem definidos, podem elucidar uma assertivo implementação. Porém, a fase teste se encontra dentro e atuante no processo de desenvolvimento sendo ativo principal na validação da qualidade e, principalmente, da eficiência. Sendo assim, através de todo o processo do desenvolvimento, desembocando no processo de testes, é possível saber se o sistema atende os requisitos, ou seja funciona de forma correta e atende a demanda do usuário.

Baseado em : Seção Quick Look - Process Models - Software Engineering A Practitioner's Approach eighth edition - Página 40.

5.3. Why does an iterative process make it easier to manage change? Is every agile process discussed in this chapter iterative? Is it possible to complete a project in just one iteration and still be agile? Explain your answers.

Segundo Pressman, um fluxo de processo iterativo repete uma ou mais das atividades antes de passar para a próxima. Sendo assim, se torna mais fácil a gerência de alterações, uma vez que, existe uma recorrência de processos permitindo que alteração aja de forma local sem prejudicar o desenvolvimento global do sistema.
Ainda segundo Pressman, *"A engenharia de software ágil combina filosofia com um conjunto de princípios de desenvolvimento. A filosofia defende a satisfação do cliente e a entrega de incremental prévio"*. Sendo assim, baseado num processo de desenvolvimento

incremental, o desenvolvimento ágil é iterativo. A entrega em apenas uma iteração vai de encontro aos princípios de agilidade apresentados por Pressman, como:

- A maior prioridade é satisfazer o cliente por meio de entrega adiantada e contínua de software valioso
- Acolha bem os pedidos de alterações, mesmo atrasados no desenvolvimento. Os processos ágeis se aproveitam de mudanças como uma vantagem competitiva na relação com o cliente.
- Entregue software em funcionamento frequentemente, de algumas semanas para alguns meses, dando preferência a intervalos mais curtos.

Retirado de : Seção 3.1 - A GENERIC PROCESS MODEL - Software Engineering A Practitioner's Approach eighth edition - Página 31.
Seção Panorama - Engenharia de Software uma Abordagem Profissional Sétima edição - Páginas 81, 84.

6.6. Which of the four organizational paradigms for teams (Section 6.4) do you think would be most effective (a) for the IT department at a major insurance company; (b) for a software engineering group at a major defense contractor; (c) for a software group that builds computer games; (d) for a major software company? Explain why you made the choices you did.

(a) e (b). closed paradigm – Por precisar de precisão nas informações e uma estrutura bem definida e esse tipo de paradigma é indicado para processos de desenvolvimento de problemas conhecidos.

(d) e (b) Open paradigm - Por que esse tipo de abordagem se propõe a trabalhar de forma colaborativa, com comunicação pesada e consenso na tomada de decisão. Além disso é indicada para a solução de problemas complexos

Retirado de : Seção 6.4 - TEAM STRUCTURES - Software Engineering A Practitioner's Approach eighth edition - Página 92.

7.9. Describe what granularity means in the context of a project schedule.
Granularidade refere-se ao nível de detalhe que é introduzido como um plano de projeto é desenvolvido. Por tanto, um contexto de projeto com o nível de granularidade alta fornece detalhes significativos da tarefa de trabalho que são planejados em incrementados em tempo relativamente curtos (para que o rastreamento e controle ocorre com frequência).

Retirado de : Seção 8.1 - REQUIREMENTS ENGINEERING - Software Engineering A Practitioner's Approach eighth edition - Página 135.

7.13. What is a successful test?

O objetivo de uma processo de Teste é, numa abordagem literal, encontrar um erro.
Por tanto, um teste bem sucedido é aquele que fornece a equipe de desenvolvimento um diagnóstico completo do funcionamento do software e, no seu melhor caso, encontra um novo erro que não for percebido pelos processos anteriores e seja possível tratá-lo e documentá-lo antes da entrega.

Retirado de : Seção 7.3.4 Construction Principles - Software Engineering A Practitioner's Approach eighth edition - Página 123.

8.3. Discuss some of the problems that occur when requirements must be elicited from three or four different customers.

A fase da engenharia de requisitos ligada a parte de negociação e elaboração é de fundamental importância para da definição do escopo do projeto. Portanto, quanto mais agentes estiverem presentes nesse processo, mais informações circularam nesse estágio o que pode levar a objetivos dúbios, conflitos de definição do problema entre outros fatores. Sendo assim, é preciso estabelecer padrão para a coleta dos requisitos visando melhor gestão de conflitos (para melhor aproveitamento do tempo), uma clara e bem definida ordem de prioridade de requisitos, proporcionando um escopo mais enxuto e assertivo.

Retirado de : Seção 7.3.4 Construction Principles - Software Engineering A Practitioner's Approach eighth edition - Páginas 121,122 e 123.

8.17. What five tasks make up a comprehensive requirements monitoring program?

(1) a depuração distribuída descobre erros e determina
minas sua causa,
(2) a verificação em tempo de execução determina se o software corresponde
sua especificação,
(3) a validação em tempo de execução avalia se o software em evolução
atende às metas do usuário,
(4) o monitoramento da atividade comercial avalia se um sistema
satisfaz as metas de negócios e
(5) evolução e design de código fornecem informações para
partes interessadas à medida que o sistema evolui.

Retirado de : Seção 8.7 - REQUIREMENTS MONITORING - Software Engineering A Practitioner's Approach eighth edition - Página 161.

10.7. What is an analysis package and how might it be used?

É uma parte importante da modelagem de análise, a qual se fundamenta na categorização dos atores envolvidos no processo de desenvolvimento. Ou seja, vários elementos do modelo de requisitos (por exemplo, casos de uso, classes de análise) são categorizados em

agrupamentos como chamados de pacote de análise usado para montar uma coleção de classes relacionadas, dando um nome representativo.
Exemplo de uso de um processo de desenvolvimento de jogo para videogame:

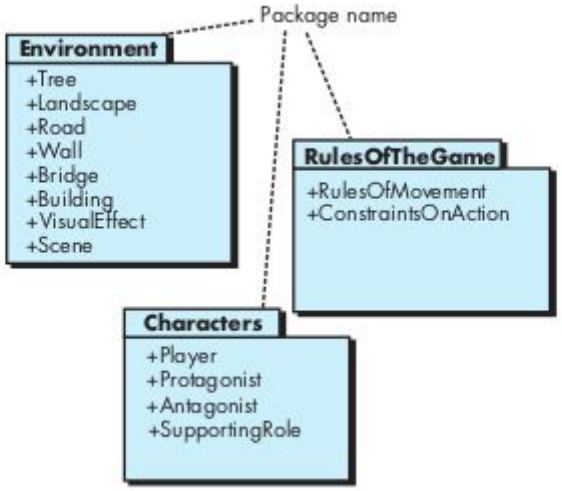

Figura 10.7

Retirado de : Seção 10.6 - Analysis Package - Software Engineering A Practitioner's Approach eighth edition - Páginas 199,200.

11.2. How does a sequence diagram differ from a state diagram? How are they similar?

O diagrama de estados é um método para representar o comportamento de um sistema, representando seus estados e os eventos que fazem com que o sistema mude de estado. Um estado é qualquer modo de comportamento observável. Além disso, o diagrama de estados indica quais ações (por exemplo, ativação do processo) são executadas como consequência de um evento específico. Por sua vez, o diagrama de sequência, indica como os eventos causam transições de objeto para objeto. Após a identificação dos eventos examinando um caso de uso, o modelador cria um diagrama de sequência - uma representação de como os eventos causam fluxo de um objeto para outro em função do tempo. Pode ser considerado como uma versão abreviada do caso de uso. Representa as principais classes e os eventos que fazem o comportamento fluir de classe para classe.

Complemento :
"Um diagrama de sequência normalmente mostra a execução de um caso de uso específico para o aplicativo e os objetos (como nas instâncias de uma classe) envolvidos na execução desse caso de uso. Ele pode mostrar um único caminho, ou todos os vários caminhos, através do caso de uso, começando com um ator (usuário, sistema externo, evento) iniciando algum tipo de ação.

Os diagramas de estados mostram os vários estados que são válidos para um objeto (que pode ser qualquer coisa, de um método a uma classe e o sistema como um todo). Essa poderia ser uma classe específica ou o sistema como um todo. Esse tipo de diagrama mostra quais ações são válidas para um determinado objeto, dependendo do estado em que ele está atualmente."
<*https://softwareengineering.stackexchange.com/questions/102346/difference-between-statechart-and-sequence-diagram*>

Retirado de : Seções 8.5.1, 11.3 - Analysis Package - Software Engineering A Practitioner's Approach eighth edition - Páginas 156,205.