Aluno: Vinícius Gomes de França

2.5 - Umbrella activities são aplicáveis em todo o processo de software, que são consideradas atividades de apoio, são elas: Controle e acompanhamento do projeto, Administração de riscos, Garantia da qualidade de software, Revisões técnicas, Medição, Gerenciamento da configuração de software, Gerenciamento da reusabilidade, Preparo e produção de artefatos de software. Cada projeto, analisará a necessidade e terá assim conhecimento do que se mais precisa e em que parte do processo será aplicada.
Ou seja, o processo adotado para um determinado projeto, pode ser muito diferente daquele adotado para outro. Algumas diferenças são: Fluxo geral de atividades, ações e tarefas e suas interdependências, Grau de prescrição da organização da equipe, nível de autonomia dada à equipe de software e o modo de aplicar as atividades de acompanhamento e controle do projeto, etc.

4.10 - O problema é que cada modelo de processo de software serve para um tipo de aplicabilidade diferente. Temos maneiras de como conseguir construir um software corretamente, mas nao implica que nosso processo construtivo seguirá fielmente aquilo. Temos modelos de processo prescritivos, modelos de processo sequenciais (cascata e movelo V), modelos de processo incremental, modelo de processo concorrente, model pessoal e o de equipe. O que podemos afirmar é que a perfeita aplicabilidade do modelo é de extrema dificuldade devido a tantas variáveis no processo e projeto e por isso não conseguimos sempre construir um software perfeito.

5.3 Porque na economia moderna, as condições de mercado mudam rapidamente, tanto o cliente quanto o usuário final está em constante mutabilidade. O Processo ágil, faz com que o software seja capaz de se adaptar e manipular, sem excessos de maneira ágil ao mundo de negócios. Sim, todos são iterativos para que possam sempre se adaptar. Não, pois o processo ágil é justamente necessário a iteração para lidar com as mudanças.

7.9 granularidade é referente a particularidades que referem-se ao nível de detalhamento introduzido conforme o plano de projeto.
Um alto grau de granularidade fornece considerável detalhamento de tarefas planejadas para incrementos em intervalos relativamente curtos para que o rastreamento e controle ocorram com frequência.
Um plano com baixo grau de granularidade resulta em tarefas mais amplas para intervalos maiores.
No geral, há variações de granularidade de acordo com o cronograma de projeto se distancia da data corrente

7.13 O teste consiste em um processo de execução de um programa com o intuito de encontrar um erro.
Um teste bem-sucedido é aquele que descobre um erro ainda não descoberto.

8.3 Problemas de escopo. Os limites do sistema são definidos de forma precária ou os cliente/usuários especificam detalhes técnico desnecessários que podem confundir, em vez de esclarecer, os objetivos globais do sistema.

Problemas de entendimento. Os cliente/usuários não estão completamente certos do que é preciso, têm um entendimento inadequado das capacidades e limitações de seus ambientes computacionais, não possuem um entendimento completo do domínio do problema, têm problemas para transmitir suas necessidades ao engenheiro do sistema e omitem informações que acreditam ser "óbvias".

Problemas de volatilidade. Os requisitos mudam com o tempo.

Ou seja, se já temos que nos preocupar com todos esses problemas quando é exclusivo para um cliente, quando se tem três ou mais, tende a ficar mais amplo e fácil de encontrar problemas de ambiguidades ou informações difusas.

8.17
Os cincos testes: (1) distributed debugging, (2) run-time verification, (3) run-time validation, (4) business activity monotiring, (5) evolution and codesign.

10.7 Pacotes de análises serve para a categorização. Vários elementos do modelo de análise (casos de uso, classes de análise) são categorizados de modo que são empacotados como um agrupamento que recebe um nome representativo.
Por exemplo, um pacote de análise para um software de jogo seria "Ambiente do jogo" onde alocaria classes como "Árvore, paisagem, rodovia, parede" outro seria "Personagens de jogo" onde alocaria classes como "protagonista, antagonista, apoios do protagonista".

11.2 Diagramas de estados é um diagrama de estados UML que representa estados ativos para cada uma das classes e para os eventos que provocam mudanças entre esses estados ativos.
Diagramas de sequência indica como os eventos provocam transições de objeto para objeto. Uma vez que os eventos tenham sido identificados pelo exame de um caso de uso. o modelador cria um diagrama de sequencia que é uma representação de como os eventos provocam o fluxo de um obejto para outor em função do tempo. Em resumo, o diagrama de sequencia é uma versão abreviada do caso de uso.

A semelhança entre eles é que ambos atuam sobre o estado do objeto/classe.