UNIVERSIDADE ESTADUAL DO CEARÁ – UECE
CENTRO DE CIÊNCIAS E TECNOLOGIAS - CCT
CURSO DE LICENCIATURA EM COMPUTAÇÃO
PÓLO: MAURITI
DISCIPLINA: ENGENHARIA DE SOFTWARE
PROF. MARCOS EDUARDO DA SILVA SANTOS
ALUNO: GABRIEL GOMES

Atividade 1:Engenharia de Software

Mauriti-CE
2019

1°)

Foi nome dado diante das dificuldades que surgiu nos anos 70, O termo expressava as dificuldades do desenvolvimento de software frente ao rápido crescimento da demanda por software, da complexidade dos problemas a serem resolvidos e da inexistência de técnicas estabelecidas para o desenvolvimento de sistemas que funcionassem adequadamente ou pudessem ser validados. No Início dos anos 70, quando vivia-se a terceira era do software, houveram muitos problemas de prazo e custo no desenvolvimento de software, devido a baixa produtividade, baixa qualidade e difícil manutenção do software. Grande parte dos projetos continuam com estes problemas ainda na atualidade, assim pode-se dizer que a crise continua vigente, A criação da Engenharia de Software surgiu numa tentativa de contornar a crise do software e dar um tratamento de engenharia(mais sistemático e controlado) ao desenvolvimento de sistemas de software complexos.

2°) E uma área do conhecimento da Computação que busca estruturar de forma racional e científica, através do uso de modelos matemáticos, a especificação, desenvolvimento e manutenção de sistemas de software**,** entre outras coisas, garante a consistência da execução desses passos, aplicando técnicas comprovadamente eficientes em cada uma delas.

3°) Coesão e Acoplamento são princípios de engenharia de software muito utilizados. Quando queremos ter uma arquitetura madura e sustentável, temos que levar em conta estes dois princípios, pois cada um deles tem um propósito específico que visa melhorar o design do software. O que acontece é que muitas pessoas não sabem a diferença entre eles e acabam não conseguindo obter os benefícios que colocá-los em prática na hora de se desenhar a arquitetura de um software.

Coesão está, na verdade, ligado ao princípio da responsabilidade única, que foi introduzido por Robert C. Martin no inicio dos anos 2000 e diz que uma classe deve ter apenas uma única responsabilidade e realizá-la de maneira satisfatória, ou seja, uma classe não deve assumir responsabilidades que não são suas.

Já o acoplamento significa o quanto uma classe depende da outra para funcionar. E quanto maior for esta dependência entre ambas, dizemos que estas classes elas estão fortemente acopladas. O forte acoplamento também nos traz muitos problemas, problemas até semelhantes aos que um cenário pouco coeso nos traz.

O desenvolvimento de softwares com alta coesão e fraco acoplamento facilita, entre outras coisas, a manutenção e o reuso. Assim sendo, o estudo desses conceitos e seus desdobramentos torna-se importante para melhorar a qualidade de sistemas. [/lead]

Muitos desenvolvedores de software percorreram caminhos que passaram pelo desenvolvimento procedural antes de chegarem à orientação a objetos. Desenvolver um programa procedural significa entender e conceber o programa como um conjunto de procedimentos (também conhecidos como rotinas, sub-rotinas ou funções) que manipulam dados. Neste modelo de desenvolvimento, os procedimentos são geralmente as unidades de subdivisão ou modularidade de sistemas.

4°) Alternativa (A) Secundária

Alternativa (B) Secundária

5°) As atividades principais envolvidas incluem o Projeto da arquitetura e o Projeto detalhado.

No projeto arquitetônico são abordados os aspectos estratégicos de projeto externo e interno, definindo a divisão do produto em subsistemas e escolhendo as tecnologias mais adequadas. Esta divisão visa facilitar o trabalho de implementação uma vez que promove o reuso.

O projeto detalhado envolve uma análise detalhada dos artefatos gerados nas atividades anteriores, de forma a construir o conhecimento necessário para a implementação do sistema. Em particular, o detalhamento dos casos de uso visa à resolução de detalhes dos fluxos envolvidos, considerando os componentes, suas interfaces e os relacionamentos entre eles.

6°) o objetivo da validação e da verificação é assegurar que o SW seja adequado e se atende às necessidades, ou seja, a confirmação de que este cumpra suas especificações.

A Verificação é uma atividade, a qual envolve a análise de um sistema para certificar se este atende aos requisitos funcionais e não funcionais. Já a Validação, é a certificação de que o sistema atende as necessidades e expectativas do cliente. O processo de Validação e Verificação, não são processos separados e independentes.

7°) O modelo incremental combina elementos do modelo cascata sendo aplicado de maneira interativa. O modelo de processo incremental é interativo igual à prototipagem, mais diferente a prototipagem o incremental tem como objetivo apresentar um produto operacional a cada incremento realizado. Esse modelo é muito útil quando a empresa não possui mão de obra disponível no momento para uma implementação completa, dentro do prazo estipulado. O modelo incremental é baseado, de forma sequencial, no modelo cascata onde se pode fragmentar as atividades e obter um sucesso maior ao analisarmos os erros, riscos, possibilitando mudanças ao longo do desenvolvimento, permitindo revisões e alterações em suas fases. Ao aplicar o modelo incremental, pode-se interagir melhor com o desenvolvimento fazendo a equipe de desenvolvedores participantes de forma ativa de modo que possa aprender e crescer no decorrer das mudanças, no qual podemos acomodar os dados e ao mesmo tempo entregar partes desse incremento ao cliente, se assim desejar. Porém, é necessário um grande esforço por parte de seus estágios, sendo bem planejados e, esse planejamento deve ser feito com um bom gerenciamento, sempre buscando identificar possíveis erros e riscos para produzir um software confiável dentro do cronograma. Da mesma forma que o modelo cascata, a comunicação é essencial para o sucesso do desenvolvimento do modelo incremental.