Estratégia Concursos

Aula 00

Passo de Tecnologia da Informação p/ TRF 3ª Região (Técnico Jud - Informática) 2021 - Pré-Edital

Autor:
Thiago Rodrigues Cavalcanti

01 de Abril de 2021

APRESENTAÇÃO DO PROFESSOR

Olá Senhoras e Senhores,

Eu me chamo Thiago Cavalcanti. Sou funcionário do Banco Central do Brasil, passei no concurso em 2010 para Analista de Tecnologia da Informação (TI). Atualmente estou de licença, cursando doutorado em economia na UnB. Também trabalho como professor de TI no Estratégia e participo da equipe do Passo Estratégico como Analista de Informática.

Tenho graduação em Ciência da Computação pela UFPE e mestrado em Engenharia de Software. Já fui aprovado em diversos concursos tais como ANAC, BNDES, TCE-RN, INFRAERO e, claro, Banco Central. A minha trajetória como concurseiro durou pouco mais de dois anos. Neste intervalo, aprendi muito e vou tentar passar um pouco desta minha experiência ao longo deste curso.

A banca do concurso é a Fundação Carlos Chagas - FCC. O concurso em questão é do **Tribunal Regional Federal da 3ª Região**. Teremos muito trabalho pela frente, tendo em vista que o assunto é extenso e o prazo é curto, além de assuntos pouco convencionais que fazem do conteúdo

programático do edital. Esteja atento, pois, a FCC consegue explorar o conteúdo com questões condizentes com o conhecimento exigido para o trabalho no órgão público.

1 – INTRODUÇÃO

Essa é a primeira aula do nosso curso e nela faremos uma apresentação teórica sobre os conceitos de engenharia de software. Dentro dos conceitos de engenharia de software vamos ver o processo de desenvolvimento de software. Leia esta aula com atenção e caso haja alguma dúvida, não hesite em me perguntar no fórum.

2 – ANÁLISE ESTATÍSTICA

Realizamos a análise estatística dos assuntos mais cobrados pela banca nas últimas provas. Diante disso, foi possível concluir que a porcentagem de questões do conteúdo dessa aula nas últimas provas elaboradas pela banca é de **9,38%** de um total de 1045 questões de todo conteúdo. Diante disso, podemos destacar que o assunto "**Conteúdo Desenvolvimento: Processo de desenvolvimento de software: conceitos básicos (CMMI, NBR/ISO 12207), UML e MPS.BR**" possui grau de importância **médio**.

Como o percentual de cobrança de cada assunto pode sofrer grandes variações, vamos classificar a importância de cada tema nos seguintes grupos:

Percentual de cobrança	Grau de importância
até 5%	baixo
de 5% a 15%	médio
acima de 15%	alto

Não posso deixar de alertá-lo que apesar do grau de importância, nenhum assunto deve ser deixado de lado. Lembre-se que as bancas sigam padrões, elas podem inovar e surpreender.

3 – ANÁLISE DE QUESTÕES

Vamos agora fazer uma análise detalhadas das questões da FCC. Cada questão servirá de ponto de partida para expandir o conhecimento relacionado. Desta forma, leia com bastante cuidado os comentários das questões. Não esqueça que os comentários complementam a parte teórica da aula.

1. (FCC / TRT - 2ª REGIÃO (SP) – 2018)

Considere que um Analista de TI foi designado para atuar em duas áreas de processo do CMMI-DEV v1.3 definidas no nível de maturidade 3, que visam demonstrar que um produto ou componente de um produto satisfaz seu uso pretendido quando colocado no ambiente alvo e garantir que produtos selecionados satisfaçam aos requisitos especificados para ele. O Analista atuará nas áreas de processo

a) Desenvolvimento de Requisitos e Medição e Análise.

b) Verificação e Gerenciamento de Configuração.

c) Medição e Análise e Validação.

d) Validação e Verificação.

e) Garantia de Qualidade e Validação.

Comentários

Na aula vimos que as áreas do nível 3 de maturidade do CMMI são:

- Desenvolvimento de Requisitos
- Solução Técnica
- Integração de Produto
- Verificação e Validação
- Foco nos Processos da Organização
- Definição dos Processos da Organização
- Treinamento na Organização
- Gestão Integrada de Projeto
- Gestão de Riscos
- Análise e Tomada de Decisões

Portanto, a alternativa correta é a letra D.

Gabarito: alternativa D.

---

2. (FCC / TRE-PR – 2017)

De acordo com o Modelo de Referência MPS para Software (MR-MPS-SW), alguns processos podem ser excluídos do escopo de uma avaliação MPS, total ou parcialmente, por não serem pertinentes ao negócio da unidade organizacional que está sendo avaliada. Cada exclusão deve

ser justificada no Plano de Avaliação. A aceitação das exclusões e suas justificativas é responsabilidade do Avaliador Líder. É permitida a exclusão completa do seguinte processo do nível de maturidade F, desde que não executado pela organização,

a) Desenvolvimento para Reuso – DRE.

b) Aquisição – AQU.

c) Integração do Produto – ITP.

d) Gerência de Reúso – GRE.

e) Medição de Riscos – MRI.

Comentários

De acordo com o Guia Geral MPS.BR, os processos que podem ser excluídos são:

Aquisição (AQU) nível F - Desde que não executados pela organização.

Gerência de Portfólio de Projetos (GPP) nível F - Desde que a única atividade da unidade organizacional seja evolução de produto.

Desenvolvimento para Reutilização (DRU) nível C - Desde que seguidas as orientações da tabela de oportunidade x capacidade do guia geral.

Portanto, alternativa correta, letra B.

Gabarito: alternativa B.

---

3. (FCC / TRE-PR – 2017)

Considere, por hipótese, que o Tribunal Regional Eleitoral está adotando as práticas do Modelo de Referência MPS para Software (MR-MPS-SW) do guia MPS.BR. O Tribunal já atingiu um nível de maturidade no qual os processos Gerência de Projetos e Gerência de Requisitos foram implantados, bem como os atributos de processo indicados. Em busca de mais um nível de maturidade, o Tribunal está implantando o processo Garantia da Qualidade, visando assegurar que os produtos de trabalho e a execução dos processos estejam em conformidade com os planos, procedimentos e padrões estabelecidos pela equipe de analistas. Desta forma, o Tribunal pretende atingir o nível de maturidade

a) D – Parcialmente Definido.

b) E – Gerenciado.

c) E – Parcialmente Definido.

d) F – Parcialmente Gerenciado.

e) F – Gerenciado.

Comentários

Para responder essa questão vamos relembrar os níveis de maturidade do MPS.BR.

- A (Em Otimização);
- B (Gerenciado Quantitativamente);
- C (Definido);
- D (Largamente Definido);
- E (Parcialmente Definido);
- F (Gerenciado); e
- G (Parcialmente Gerenciado).

Ou seja, a única alternativa que encaixa perfeitamente com um dos níveis é a letra E.

Gabarito: alternativa E.

---

4. (FCC / TRT - 11ª Região (AM e RR) – 2017)

O Modelo de Referência MPS para Software (MR-MPS-SW) define níveis de maturidade que são uma combinação entre processos e sua capacidade. Considere:

I. é a caracterização da habilidade do processo para alcançar os objetivos de negócio, atuais e futuros.

II. está relacionada com o atendimento aos atributos de processo associados aos processos de cada nível de maturidade.

III. expressa o grau de refinamento e institucionalização com que o processo é executado na organização ou na unidade organizacional.

IV. é representada por um conjunto de atributos de processo, dentre os quais se encontra "O processo é objeto de análise quantitativa".

Refere-se à capacidade do processo o que se afirma em

a) I e II, apenas.

b) II, III e IV, apenas.

c) III e IV, apenas.

d) I e IV, apenas.

e) I, II, III e IV.

Comentários

Note que essa questão traz 4 itens, e todos são relacionados aos termos e definições do MPS.BR.

Na aula vimos que a capacidade do processo no MPS possui os seguintes atributos de processos (AP):

1) **AP 1.1 - O processo é executado** (Nível G) - o processo realiza o que foi proposto para ele;
2) **AP 2.1 - O processo é gerenciado** (Nível G e F) - a execução do processo é gerenciada;
3) **AP 2.2 - Os produtos de trabalho do processo são gerenciados** (Nível G e F) - os produtos de trabalho realizados pelo processo são gerenciados de forma adequada;

4) **AP 3.1 - O processo é definido** (Níveis E, D e C) - existe um padrão e que o mesmo apoia a implementação do processo;
5) **AP 3.2 - O processo está implementado** (Níveis E, D e C) - o processo, padronizado, é de fato implementado para atingir os seus objetivos;
6) **AP 4.1 – O processo é medido** (Nível B) - algumas medições são usadas para assegurar que o desempenho do processo ajuda a alcançar os objetivos para o qual este processo foi proposto;
7) **AP 4.2 – O processo é controlado** (Nível B) - o processo é controlado estatisticamente, permitindo se ter previsibilidade, estabilidade e capacidade de execução;
8) **AP 5.1 – O processo é o objeto de inovações** (Nível A) - as mudanças no processo são identificadas a partir da análise dos seus indicadores e da investigação de possíveis inovações;
9) **AP 5.2 – O processo é otimizado continuamente** (Nível A) - as mudanças no processo têm, de fato, impacto no alcance dos objetivos, no que tange os aspectos relevantes de melhoria do mesmo.

Portanto, a alternativa correta é a letra E.

Gabarito: alternativa E.

---

5. FCC / TRT - 11ª Região (AM e RR) – 2017)

Considere que o TRT decidiu implantar as melhores práticas de qualidade de software de acordo com o modelo de referência MRMPS-SW do guia MPS.BR. Um Técnico indicou, corretamente, que o Tribunal deveria iniciar pelo nível

a) 0 e, à medida que evoluir nos níveis de capacidade, um maior nível de maturidade para desempenhar os processos devem ser atingidos.

b) de maturidade A e este nível não possui processos específicos.

c) de maturidade G, que é composto pelos processos Gerência de Projetos e Gerência de Requisitos.

d) de maturidade G e este nível não possui processos específicos.

e) 1 e, à medida que evoluir nos níveis de maturidade, um maior nível de capacidade para desempenhar os processos devem ser atingidos.

Comentários

O nível de maturidade G é composto pelos processos gerência de projetos e gerência de requisitos e tem, entre outros propósitos, estabelecer e manter planos que definam as atividades, os recursos e as responsabilidades do projeto. Portanto, a alternativa C é a resposta para nossa questão.

Gabarito: alternativa C.

---

6. (FCC / TRE-SP – 2017)

Considere, por hipótese, que o TRE-SP esteja em busca de uma certificação de qualidade de software. Antes da escolha, uma equipe de Analistas realizou uma comparação relativa ao

tratamento da Garantia da Qualidade entre o MR-MPS-SW e o CMMIDEV 1.3. Este comparativo indica, corretamente, que o

a) CMMI-DEV exige a identificação, registro e comunicação dos problemas e das não conformidades, relacionados à avaliação de processos e produtos.

b) MR-MPS-SW só exige o estabelecimento de registros das atividades de Garantia de Qualidade, que é parte do exigido no CMMI-DEV.

c) escalonamento das ações corretivas para níveis superiores exigido pelo CMMI-DEV, não é exigido no MR-MPS-SW, o que pode deixar a solução das não conformidades fragilizadas.

d) MR-MPS-SW exige que a avaliação da aderência dos produtos de trabalho seja realizada sempre antes da entrega ao cliente externo, bem como em marcos do projeto.

e) CMMI-DEV exige que a avaliação da aderência dos produtos de trabalho seja realizada apenas antes da entrega a um cliente interno.

Comentários

Essa questão é um dos absurdos cobrados pela FCC. Ela exige que você decore os 4 pontos da gerência de qualidade. Relembrando, eles são:

GQA 1. A aderência dos produtos de trabalho aos padrões, procedimentos e requisitos aplicáveis é avaliada objetivamente, antes dos produtos serem entregues e em marcos predefinidos ao longo do ciclo de vida do projeto;

GQA 2. A aderência dos processos executados às descrições de processo, padrões e procedimentos é avaliada objetivamente;

GQA 3. Os problemas e as não-conformidades são identificados, registrados e comunicados;

GQA 4. Ações corretivas para as não-conformidades são estabelecidas e acompanhadas até as suas efetivas conclusões. Quando necessário, o escalamento das ações corretivas para níveis superiores é realizado, de forma a garantir sua solução;

Portanto a alternativa correta é letra D.

Gabarito: alternativa D.

---

7. (FCC / TRT-23ª REGIÃO (MT) – 2016)

Para a contagem de pontos de função, um Técnico do Tribunal montou a seguinte tabela com base em seus levantamentos iniciais:

Tipo de Função	Complexidade Funcional		
		Qt	Val
ALI	Simples Média Complexa	1 2 0	7 10 15
AIE	Simples Média Complexa	2 0 0	5 7 10
Entrada	Simples Média Complexa	0 1 6	3 4 6
Saída	Simples Média Complexa	0 1 2	4 5 7
Consulta	Simples Média Complexa	1 3 1	3 4 6

Com estes dados, o total de pontos de função transacional corretamente contados por ele foi

a) 37.

b) 5.

c) 117.

d) 80.

e) 15.

Comentários

Abaixo da tabela temos a informação mais importante para responder essa questão: "o total de pontos de **função transacional**". São elas: Entrada, Saída e Consulta. O cálculo deve ser realizado apenas com os valores dessas funções.

EE = 0 x 3 + 1 x 4 + 6 x 6 = 40.

SE = 0 x 4 + 1 x 5 + 2 x 7 = 19.

CE = 1 x 3 + 3 x 4 + 1 x 6 = 21.

40 + 19 + 21 = 80.

Gabarito: alternativa D.

---

8. (FCC / Prefeitura de Teresina-PI – 2016)

A equipe de Analistas de Sistemas da PRODATER reuniu-se para escolher uma metodologia de desenvolvimento capaz de atender às seguintes características do projeto e do cliente:

– Satisfazer o cliente através da entrega contínua e adiantada de software, em períodos curtos de tempo.

– Permitir que haja mudanças nos requisitos, mesmo tardiamente ao desenvolvimento.

– Pessoas de negócio e desenvolvedores devem poder trabalhar juntos por todo o projeto.

– As equipes devem ser auto-organizáveis.

Uma escolha que atende a lista de características é a metodologia

a) RAD – Rapid and Agile Development, muito adequada para projetos que envolvem altos riscos técnicos, como novas tecnologias.

b) Espiral, mas somente se o projeto for de pequeno porte, pois o uso da prototipagem aumenta os riscos de fracasso.

c) de Desenvolvimento Concorrente, embora tenha etapas sequenciais muito próximas do modelo em cascata.

d) RUP – Rational Unisied Process, o framework ágil mais utilizado por ser o mais simples e de fácil adoção e adaptação.

e) Extreme Programming (XP), que tem aderência à orientação a objetos como um paradigma de desenvolvimento e segue o princípio KIS – Keep it Simple.

Comentários

Analisando as alternativas temos:

a) ERRADA. RAD não é recomendado para projetos que envolvem altos riscos técnicos (Pressman)
b) ERRADA. Prototipagem é muito utilizado para elicitação de requisitos e não aumenta os riscos de fracasso
c) ERRADA. Cascata é sequencial; concorrente, como o próprio nome já diz, é executado em paralelo.
d) ERRADA. RUP é um framework verboso, muita documentação, artefato, não é de simples adoção
e) CORRETA.

Gabarito: alternativa E.

---

9. (FCC / ELETROBRAS-ELETROSUL – 2016)

Atualmente os softwares podem ser desenvolvidos utilizando-se métodos ágeis ou métodos tradicionais. A escolha da metodologia mais adequada vai depender de vários fatores, como por exemplo, a característica de projeto, da empresa ou da gestão. Para fazer a escolha correta, é necessário ainda conhecer as características dos principais métodos e modelos de processo de desenvolvimento de software. Sobre estes métodos e modelos de processo é correto afirmar:

a) As metodologias ágeis são indicadas principalmente em casos em que os requisitos são bem compreendidos e provavelmente não sofrerão grandes alterações durante o desenvolvimento do sistema.

b) Os diagramas de Caso de Uso da UML são utilizados intensamente na fase de Elaboração do Rational Unified Process – RUP para criar um modelo de requisitos para o sistema.

c) Nos modelos em cascata os testes são desenvolvidos paralelamente aos requisitos, antes de iniciar o desenvolvimento, ajudando testadores e desenvolvedores a compreenderem os requisitos.

d) No Rational Unified Process – RUP o cliente participa do processo de desenvolvimento discutindo cenários com a equipe para gerar os cartões de estórias, que englobam as necessidades do cliente.

e) Sprinter e programação em pares são práticas descritas e amplamente utilizadas na eXtreme Programming – XP para agilizar o processo de desenvolvimento e reduzir a possibilidade de erros.

Comentários

Analisando as alternativas temos:

a) ERRADA. O indicado nesse caso é o modelo cascata.
b) CORRETA.
c) ERRADA. Não há paralelismo no modelo cascata. Todas as fases são sequenciais.
d) ERRADA. Cartões de estórias fazem parte do modelo XP.
e) ERRADA. Sprint é no modelo Scrum.

Gabarito: alternativa B.

---

10. (FCC / PGE-MT – 2016)

Considere que uma organização está no nível de maturidade G do modelo MR-MPS-SW, do guia MPS.BR. Este nível é composto por 2 processos. Esta organização

a) encontra-se no nível Parcialmente Definido, o mais alto da escala, compatível com o nível de maturidade 5 do CMMI-DEV versão 1.3.

b) deve cumprir o atributo de processo cujo resultado da sua implementação completa é: os processos que estão alinhados a objetivos quantitativos de negócio são identificados.

c) deve cumprir o atributo de processo cujo resultado da sua implementação completa é: técnicas para análise dos dados coletados são selecionadas.

d) deve implementar o processo Garantia da Qualidade, cujo propósito é assegurar que os produtos de trabalho e a execução dos processos estejam em conformidade com os planos e padrões estabelecidos.

e) deve implementar o processo Gerência de Requisitos, cujo propósito inclui gerenciar os requisitos do produto e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto.

Comentários

No MPS.BR, os níveis de maturidade são avaliados a partir de atributos de processo (AP). Cada atributo de processo é detalhado por um ou mais resultados esperados (RAP). O atributo de

processo AP 1.1 (o processo é executado) evidencia o quanto o processo atinge seu propósito. Este atributo possui apenas a descrição do resultado esperado RAP 1, que traz o texto:

a) existe uma política organizacional estabelecida e mantida para o processo.

b) a execução do processo é planejada.

c) a execução do processo é monitorada e ajustes são realizados.

d) o processo atinge seus resultados definidos.

e) as informações e os recursos necessários para a execução do processo são identificados e disponibilizados.

Portanto, a alternativa correta é a letra E.

Gabarito: alternativa E.

---

11. (FCC / TRT - 20ª REGIÃO (SE) – 2016)

Um técnico trabalha em uma organização que atingiu o nível de maturidade 4 do CMMI (quantitativamente gerenciado). Para atingir este nível, todas as áreas de processo dos níveis de maturidade anteriores e as áreas de processo do nível de maturidade atual precisam atingir o nível de capacidade

a) 5 (avançado).

b) 2 (gerenciado).

c) 3 (realizado).

d) 3 (definido).

e) 4 (otimizado).

Comentários

Relembrando que os Níveis de Maturidade (CMMI v. 1.3 - Abordagem por estágios) são:

Nível 1: Inicial

Nível 2: Gerenciado

Nível 3: Definido

Nível 4. Quantitativamente gerenciado

Nível 5: Em otimização

Portanto, alternativa correta, letra D.

Gabarito: alternativa D.

---

12. (FCC / Prefeitura de Teresina - PI – 2016)

O nível G de maturidade do MR-MPS-SV é composto pelos processos Entrega de Serviços, Gerência de Incidentes, Gerência de Nível de Serviço, Gerência de Requisitos e Gerência de

Trabalhos. Neste nível, a implementação dos processos deve satisfazer os atributos de processo:

a) AP 1.1 (o processo é executado) e AP 2.1 (o processo é gerenciado).

b) AP 1.1 (o processo é medido) e AP 2.1 (o processo é definido).

c) AP 1.1 (o processo está implementado) e AP 2.1 (o processo é otimizado continuamente).

d) AP 1.1 (o processo é objeto de melhorias incrementais e inovações) e AP 2.1 (o processo é otimizado continuamente).

e) AP 1.1 (o processo é definido) e AP 2.1 (o processo está implementado).

Comentários

Na aula vimos os atributos de processos (AP) relacionados com os níveis de maturidade. Entre eles temos os que estão relacionados ao nível G de maturidade:

AP 1.1 - O processo é executado (Nível G) - o processo realiza o que foi proposto para ele.

AP 2.1 - O processo é gerenciado (Nível G e F) - a execução do processo é gerenciada.

AP 2.2 - Os produtos de trabalho do processo são gerenciados (Nível G e F) - os produtos de trabalho realizados pelo processo são gerenciados de forma adequada.

Entre as alternativas, a única que satisfaz essa relação é a letra A.

Gabarito: alternativa A.

---

13. (FCC / TRT - 14ª Região (RO e AC) – 2016)

O modelo de qualidade de software CMMI, versão 1.3,

a) é largamente utilizado pelos Tribunais que utilizam o Scrum, pois o modelo MPS.BR conflita com as metodologias ágeis de software.

b) adota a ITIL v3 como padrão de gerenciamento da qualidade de serviços de TI.

c) utiliza a representação contínua para permitir que a organização atinja níveis de capacidade.

d) não atende às necessidades das grandes empresas brasileiras, por isso foi criado o modelo MPS.BR.

e) foi modificado em 2010 para manter compatibilidade com a Norma NBR ISO/IEC 12207:2009.

Comentários

O modelo de qualidade de software CMMI, em sua versão 1.3, ***utiliza as representações contínua e por estágio para permitir que a organização atinja níveis de capacidade*** e maturidade respectivamente. Ademais, o modelo MPS.BR ***NÃO conflita com as metodologias ágeis de software***, sobretudo com o ***SCRUM***, devendo a organização realizar algumas adaptações nas práticas do Scrum para serem completamente aderentes ao MPS.BR. O CMMI ***NÃO*** adota a ITIL v3 como padrão de

gerenciamento da qualidade de serviços de TI. Além disso, ***atende às necessidades das grandes empresas brasileiras***, apesar de ser mais caro que o modelo MPS.BR. Por fim, vale lembrar que o modelo modificado em 2010 para manter compatibilidade com a Norma NBR ISO/IEC 12207:2009 foi o ***MPS.BR***, e não o CMMI.
Portanto, a alternativa que responde corretamente a questão é a letra C.

Gabarito: alternativa C.

---

14. (FCC / TRT - 14ª Região (RO e AC) – 2016)

O modelo de qualidade de software MPS.BR

a) não é compatível com a Norma NBR ISO/IEC 12207:2009.

b) adota a ITIL v3 como padrão de gerenciamento da qualidade de serviços de TI.

c) não pode ser aplicado em nenhum Tribunal Regional do Trabalho.

d) não atende às necessidades das empresas brasileiras, que preferem a adoção do CMMI.

e) não conflita com as metodologias ágeis de software, podendo ser utilizado junto com o Scrum.

Comentários

Analisando as alternativas temos:

a) Vimos na aula que uma das bases é esta norma. ERRADA.

b) Não adota nenhum padrão desta natureza. ERRADA.

c) Pode ser aplicado em qualquer organização. ERRADA.

d) Foi criado pensando nas empresas brasileiras, tanto que o MPS.BR é acrônimo de “Melhoria de Processos do Software Brasileiro”. ERRADA.

e) CORRETA.

Gabarito: alternativa E.

---

15. (FCC / TRE-RR – 2015)

Dentre os princípios da Engenharia de Software NÃO se encontra:

a) Um sistema de software existe por uma razão: para fornecer valor aos seus usuários. Todas as decisões devem ser tomadas com este princípio em mente.

b) Keep it Simple: todo projeto deve ser tão simples quanto possível, mas simples não significa rápido e mal feito. Simplificar precisa de muito raciocínio e trabalho em várias interações.

c) Sempre especifique, projete e implemente sabendo que mais alguém terá de entender o que se está fazendo. Mais alguém irá usar, manter, documentar ou precisará entender o sistema que uma pessoa desenvolve.

d) Reuso sempre poupa tempo e esforço, assim, conseguir um alto nível de reuso é a principal meta a ser alcançada no desenvolvimento. O reuso torna mais barato o custo do sistema, pois produz componentes reusáveis.

e) Raciocinar clara e completamente antes da ação quase sempre produz os melhores resultados. Quando o raciocínio límpido é aplicado ao sistema, é mais provável que funcione adequadamente.

Comentários

Antes de responder a questão, vou apresentar um "macete" para lembrar os 7 princípios de Engenharia de Software, segundo Pressman:

1) a razão de existir - O Software de existir por um motivo
2) KISS (Keep It Simple, Stupid!, ou seja: não complique!)
3) mantenha a visão - Devendo ser clara e consistente
4) o que um produz outros consomem - Documentação e Clareza
5) esteja aberto para o futuro - Desenvolva pensando na Evolução
6) planeje com antecedência, visando a reutilização
7) pense! - Depois faça

Agora com essa informação fica mais fácil de responder à questão.

A alternativa que responde à questão é a letra D.

Gabarito: alternativa D.

---

16. (FCC / TRE-AP – 2015)

A fase de projeto de software possui duas atividades básicas: projeto da arquitetura e projeto detalhado. Nesta fase

a) o sistema é codificado, a partir da descrição computacional do sistema, em uma linguagem que torna possível a compilação e a geração do código-executável para o software.

b) em um processo de desenvolvimento orientado a objetos, são criadas as classes de objetos do sistema utilizando-se ferramentas CASE e bibliotecas de classes preexistentes para agilizar a implementação.

c) em um processo de desenvolvimento orientado a objetos, o projeto da arquitetura visa distribuir as classes de objetos relacionadas do sistema em subsistemas e seus componentes, distribuindo-os pelos recursos de hardware disponíveis.

d) os diversos módulos do sistema são integrados, resultando no produto de software.

e) o projeto de arquitetura realiza a modelagem das relações de cada módulo do sistema, com o objetivo de implantar as suas funcionalidades. Além disso, são implementados os projetos de interface com o usuário e de banco de dados.

Comentários

O projeto possui duas atividades básicas: projeto da arquitetura (ou projeto de alto nível), e projeto detalhado (ou projeto de baixo nível). Em um processo de desenvolvimento orientado a objetos, o projeto da arquitetura normalmente é realizado por um arquiteto de software. O projeto da arquitetura visa distribuir as classes de objetos relacionados do sistema em subsistemas e seus componentes, distribuindo também esses componentes pelos recursos de hardware disponíveis. Já no projeto detalhado, são modeladas as relações de cada módulo com o objetivo de realizar as funcionalidades do módulo. Além de desenvolver o projeto de interface com o usuário e o projeto de banco de dados. Portanto, a alternativa correta é a letra C.

Gabarito: alternativa C.

---

17. (FCC / TRE-PB – 2015)

Análise de Pontos de Função − APF é uma técnica para medir o tamanho funcional de um software cujo processo de medição envolve diversas etapas, dentre elas, a medição das funções de dados, que envolvem as funcionalidades fornecidas pelo sistema ao usuário para atender a suas necessidades de armazenamento de dados. Dentre as funções de dados estão

a) os Arquivos de Ponto de Controle − APC.

b) as Saídas Externas − SE.

c) as Entradas Externas − EE.

d) os Arquivos de Interface Externa − AIE.

e) as Consultas Externas − CE.

Comentários

Na aula expliquei que existem dois grupos de componentes de pontos de função. São eles: Função do tipo dado que inclui: os Arquivos Lógicos Internos (ALI) e os Arquivos de Interface Externa (AIE); e Função do tipo transação que inclui: Consulta Externa (CE), Saída Externa (SE) e Entrada Externa (EE). Logo as alternativas B, C e E que fazem referência a funções do tipo transação, são descartadas. Nossa resposta está na letra D - Arquivo de Interface Externa (AIE).

Gabarito: alternativa D.

---

18. (FCC / TRT-9ª REGIÃO (PR) – 2015)

A Análise por Pontos de Função é uma técnica paramétrica para estimativa de esforço para desenvolvimento de software. Esta técnica

a) é baseada no número de linhas de código produzidas.

b) é aplicável antes dos requisitos funcionais terem sido definidos.

c) não pode ser usada para calcular o custo-benefício de software comprado.

d) é normatizada internacionalmente pelo International Function Point Users Group − IFPUG.

e) não é adequada para medir a produtividade de uma equipe de desenvolvimento.

Comentários

A Análise por Pontos de Função foi proposta por Allan Albrecht em 1979 e formalizado em 1984 pela IBM, e em 1986 foi criado o IFPUG (International Function Points Users Group). Portanto, a alternativa correta é a letra D.

Gabarito: alternativa D.

---

19. (FCC / SABESP – 2014)

Um processo de engenharia de software é formado por um conjunto de passos parcialmente ordenados, relacionados com artefatos, pessoas, recursos, estruturas organizacionais e restrições, tendo como objetivo produzir e manter os produtos de software finais requeridos. Sobre estes processos é INCORRETO afirmar que

a) usualmente considera-se que a primeira grande divisão de um processo é a fase, que consiste em um período de tempo no qual determinadas atividades com objetivos bem específicos são realizadas.

b) a maioria dos processos de software é organizada em torno de tarefas (às vezes também chamadas de atividades). Toda atividade tem um objetivo principal estabelecido e visa criar ou produzir uma mudança de estado visível em um ou mais artefatos durante a execução de um projeto.

c) todos os modelos de processo têm fases cíclicas, ou seja, o desenvolvimento passa repetidamente de uma fase para a outra, formando um ciclo repetitivo de fases até a finalização do projeto.

d) na descrição de um processo, as atividades devem ser atribuídas a perfis ou cargos, e não a pessoas. Apenas quando o processo for usado em um projeto concreto é que deve haver atribuições de atividades a pessoas.

e) um modelo de processo é um conjunto de regras mais abstratas que especificam a forma geral de processos. Apresenta uma filosofia e uma forma geral de comportamento com base na qual processos específicos devem ser definidos.

Comentários

Entre as alternativas a única que não condiz com os processos de software é a letra C. Porque por exemplo o modelo em cascata é um exemplo de modelo sequencial e linear, invalidando assim a afirmativa de que “todos os modelos de processo têm fases cíclicas”.

Gabarito: alternativa C.

---

20. (FCC / TJ-AP – 2014)

Na Análise de Pontos de Função, são contados diversos componentes, dentre os quais NÃO se encontra(m):

a) Arquivos Lógicos Internos, que correspondem aos arquivos mantidos e utilizados pelo sistema sendo contado.

b) Arquivos de Interface Externa, que correspondem aos arquivos utilizados pelo sistema sendo contado, porém mantidos por outros sistemas.

c) Entradas Externas, que correspondem a transações cujo objetivo é a manutenção de arquivos ou a alteração do comportamento do sistema.

d) Consultas Externas, que correspondem a transações cujo objetivo é a apresentação de informações aos usuários, provenientes dos arquivos, sem a geração de dados derivados, atualização de arquivos ou a utilização de cálculos/fórmulas.

e) Esforço de Desenvolvimento, que corresponde ao tamanho funcional de um software e ao esforço gasto no seu desenvolvimento, medido em pessoas-hora, resultando na quantidade de horas e, consequentemente, no preço estimado de desenvolvimento do sistema.

Comentários

Entre as alternativas a única que apresenta componentes que não são contados na Análise de Pontos de Função é a letra E. Os requisitos não-funcionais (restrições) não são levados em conta em uma contagem não-ajustada justamente porque a APF mede as funcionalidades do software.

Gabarito: alternativa E.

4 – Orientações de Revisão e Pontos a Destacar

O Processo de Desenvolvimento de Software é abordado como um conceito dentro de Engenharia de Software. Por isso, vamos destacar tópicos desse tema para termos base para nossa aula.

Conceitos de Engenharia de Software

O termo **Engenharia de Software** foi criado na década de 1960 e utilizado oficialmente em 1968 na *NATO Conference on Software Engineering* (Conferência sobre Engenharia de Software da OTAN). Ele foi criado na tentativa de contornar a dificuldade do desenvolvimento de softwares frente ao crescimento da demanda, conhecida na década de 1970 como crise do software, além de trazer uma abordagem de engenharia para o desenvolvimento de sistemas complexos.

Segundo Friedrich Ludwig Bauer[1], "Engenharia de software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais".

[1] Cientista da computação alemão, autor da primeira definição de engenharia de software.

De acordo com o IEEE[2], "Engenharia de software é a aplicação de uma abordagem sistemática, disciplinada e quantificável, para o desenvolvimento, operação e manutenção do software; isto é, a aplicação de engenharia ao software.".

Engenharia de software é uma abordagem sistemática e disciplinada para o desenvolvimento de software (PRESSMAN[3], 2006).

Em suma, a Engenharia de Software é 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 aplicando tecnologias e métodos da Ciência da Computação, Gerência de projetos, das Engenharias e outros campos do conhecimento.

A partir destes conceitos, chegamos à conclusão que Engenharia de Software não é simplesmente programação.

DEFINIÇÃO DE SOFTWARE

Em uma visão restritiva, o termo software é associado aos programas / aplicativos de computador. Entretanto, seu conceito vai muito mais além. O software são códigos em linguagem natural ou não, entendíveis por um sistema computacional, assim como qualquer dado que faça com que opere corretamente. Ou seja, ele não é apenas o programa, mas sim todos os dados e configurações necessários para que o programa funcione corretamente.

O software é o conjunto de vários artefatos e não apenas o código fonte (SOMMERVILLE, 2003).

De acordo Pressman (2006) os softwares estão divididos nas seguintes categorias:

- **Software de sistema**

São programas que apoiam outros programas, como por exemplo o software que realiza a comunicação com o hardware (sistema operacional) e software que ajuda na construção de outro software (compiladores).

[2] Instituto de Engenheiros Eletricistas e Eletrônicos (http://www.ieee.org.br/) - sociedade técnico-profissional internacional, dedicada ao avanço da teoria e prática da engenharia nos campos da eletricidade, eletrônica e computação.

[3] Roger S. Pressman é um engenheiro de software, escritor e consultor, norte-americano. Autoridade reconhecida internacionalmente nas tecnologias em melhoria de processos de software e engenharia de software.

- **Software de aplicação**

São os programas desenvolvidos para auxiliar na execução de funções específicas para o negócio de uma empresa, por exemplo o processamento de dados comerciais ou técnicos.

- **Software cientifico e de engenharia**

São programas que auxiliam no processamento de números em massa, por exemplo na astronomia, meteorologia, biologia molecular.

- **Software embutido (embarcado)**

São programas desenvolvidos para serem executados dentro de produtos específicos, como por exemplo o sistema de teclas digitais de um forno micro-ondas.

- **Software para linhas de produtos**

São conhecidos como software de prateleira, e projetados para prover capacidade específica de utilização por muitos clientes.

- **Aplicações Web / aplicativos móveis**

São programas voltados para as redes e abrangem uma série de variedades, contemplando aplicativos para dispositivos móveis.

- **Software de inteligência artificial**

São softwares que utilizam algoritmos não numéricos para solucionar problemas complexos, não passíveis de computação ou análise direta. Este tipo software está presente na robótica, no reconhecimento de voz e imagem, nos jogos.

- **Software legado**

São os sistemas de uma organização que, apesar de serem bastante antigos, fornecem serviços essenciais e geralmente utilizam bancos de dados obsoletos. Esses sistemas podem sofrer com dois extremos, ou são continuamente modificados para se adequar às mudanças, ou nunca são alterados quando atendem as necessidades do cliente. A grande preocupação em relação a eles está na baixa qualidade, pois muitas vezes não existem documentações, ou se existem são pobres de detalhes.

Software Livre

Também existe o conceito de software livre, que remete ao programa que dá liberdade ao utilizador, permitindo que ele o estude, modifique e compartilhe com outras pessoas. Para tanto, é necessário que o utilizador tenha acesso ao código-fonte, para altera-lo de acordo com as suas necessidades.

Por conta da confusão em relação a este termo, onde empresas proprietárias de softwares utilizam para associa-lo a cópia disponibilizada sem nenhum custo, o software livre também é dividido em categorias:

- **Software com copyleft**

É um software livre cujos termos de distribuição não permitem que os redistribuidores incluam restrições adicionais quando redistribuem ou modificam o software. Isso significa que toda cópia, mesmo modificada, precisa ser software livre.

- **Software livre sem copyleft**

É diferente da categoria anterior, pois o autor permite que a redistribuição e modificação possa conter restrições adicionais.

- **Domínio público**

Software não sujeito ao copyright. Este é um caso especial de software livre sem copyleft, onde as cópias ou versões modificadas podem não ser livres.

- **Software com licença GPL**

A licença GPL (*General Public License*) é a definição de licença de software idealizada por Richard Stallman em 1989, onde os softwares derivados de um produto originalmente licenciado pela GPL só podem ser distribuídos se utilizarem a mesma licença.

Copyleft significa o direito de permissão de cópia de uma obra por outros usuários, dando a liberdade de copiar, modificar e redistribuir, exigindo que esse direito seja mantido em todas as versões modificadas.

PRINCÍPIOS DA ENGENHARIA DE SOFTWARE

Os princípios da Engenharia de Software constituem a base dos métodos, tecnologias, metodologias e ferramentas adotadas que norteiam a prática de desenvolvimento de software. Os princípios se aplicam ao processo e ao produto de software através da adoção de métodos e técnicas. Os princípios-chave são:

- **Rigor e Formalidade**

Ao considerar que a engenharia de software deve ser realizada de maneira sistemática, o rigor é um complemento necessário a criatividade, que visa aumentar a confiança dos desenvolvedores. A formalidade é o rigor no seu nível mais elevado. Nesse princípio estão incluídas por exemplo a documentação rigorosa dos passos de desenvolvimento e gerenciamento, a análise sistemática dos dados de testes, a avalição dos prazos de entrega.

- **Separação de Interesses**

Abrange o domínio sobre a complexidade do software, separando os problemas principais e investindo esforços para solução de um de cada vez, dividindo as responsabilidades. Como exemplo temos o desenvolvimento por fases, separando os interesses por atividades e respeitando o tempo estipulado.

- **Modularidade**

Na modularidade um sistema complexo pode ser dividido em peças mais simples, denominadas módulos. Esse princípio depende do suporte da separação de interesses, para que ao lidar com um módulo específico seja possível ignorar os detalhes dos outros módulos. Como característica desse princípio, os módulos devem ter um alto nível de coesão, onde existe uma forte relação entre eles, mas com um baixo acoplamento, onde a interação entre eles seja baixa, possibilitando que eles sejam compreendidos como unidades em separado.

- **Abstração**

Esse conceito visa a identificação de aspectos importantes, ignorando os detalhes. Por exemplo, os botões de um relógio que são sua interface, entretanto eles são a abstração em relação ao ajuste da hora.

- **Antecipação a Mudanças**

Esse princípio este diretamente relacionado ao suporte a evolução de um software, considerando aspectos relacionados ao processo de evolução e compatibilidade com mudanças futuras.

- **Generalidade**

É o princípio que visa durante a resolução de um problema descobrir se ele é apenas uma instância de um problema mais abrangente, onde a solução pode ser reutilizada em outros casos.

- **Incrementação**

É relacionada a evolução de um software através de incrementos estruturados. Pode ser realizado através da entrega de subconjuntos de um sistema desde cedo, visando coletar o feedback dos usuários e adicionar funcionalidades de forma incremental.

CICLO DE VIDA DO SOFTWARE

Agora vamos iniciar nosso estudo sobre o ciclo de vida. Esse termo surgiu no contexto da biologia, onde trata do ciclo de vida de um ser vivo, ou seja, do conjunto de transformações pelas quais passam os indivíduos de uma espécie durante a vida.

Na imagem acima temos o exemplo do ciclo de vida do homem. Dessa forma, podemos extrair o conceito que o ciclo de vida trata das fases que um ser vivo passa desde o seu nascimento até a sua morte e aplicar esse conceito em diversos contextos.

Na Engenharia de Software esse termo normalmente é aplicado a sistemas artificias com o significado de mudanças que acontecem na vida de um produto de software e segue o mesmo princípio do conceito de ciclo de vida de um ser vivo.

Steve McConnell afirma que um modelo de ciclo de vida do software é uma representação que descreve todas as atividades que tratam da criação de um produto de software.

O ciclo de vida é a estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema, desde a definição de seus requisitos até o término de seu uso.

Portanto, podemos concluir que um ciclo de vida de software se refere às fases pelas quais um sistema de software atravessa desde sua concepção até sua retirada de produção. Esse conceito é simples de entender, o problema em relação ao ciclo de vida do software, de acordo com os autores, está nas dezenas e diferentes fases para os ciclos de vida.

Infelizmente não há consenso entre os autores e além disso, alguns deles com o passar do tempo mudam de opinião. Portanto, aqui chegamos a minha sugestão após analisar as últimas provas da

FCC em relação ao custo/benefício de estudar uma vasta lista de vários modelos de ciclos de vida do software, é muito alto. Sugiro que você guarde espaço na memória para assuntos que são mais recorrentes. De toda forma vou citar alguns modelos de ciclos de vida do software para você compreender como eles funcionam e as diferenças entre eles.

O modelo de ciclo de vida é a primeira escolha a ser feita no processo de software. A partir desta escolha será definida desde a maneira mais adequada de obter as necessidades do cliente, até quando e como o cliente receberá sua primeira versão operacional do sistema.

Diante disso, vou apresentar os seguintes modelos:

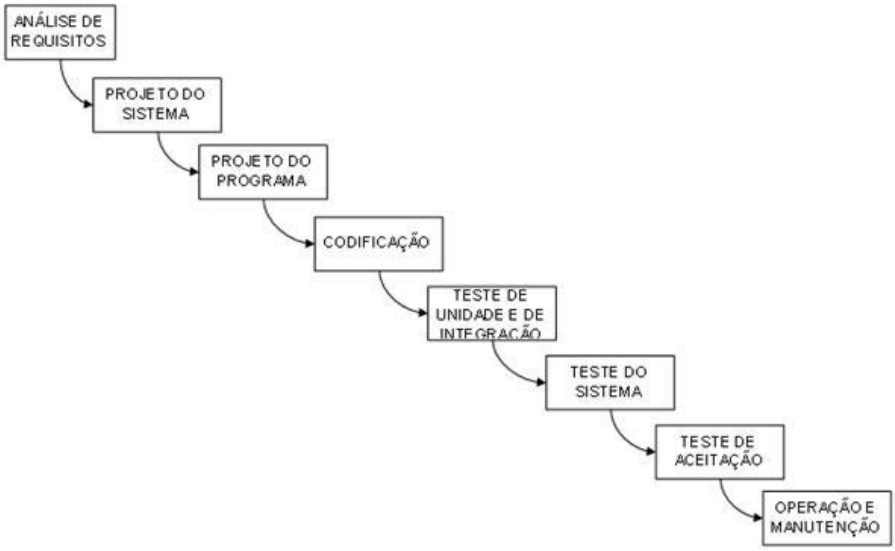
- Cascata
- Modelo em V
- Incremental
- RAD - *Rapid Application Development*
- Prototipagem

Modelo em Cascata

Formalizado por Royce em 1970, é o modelo mais antigo. Suas atividades fundamentais são:

- análise e definição de requisitos;
- projeto;
- implementação;
- teste;
- integração.

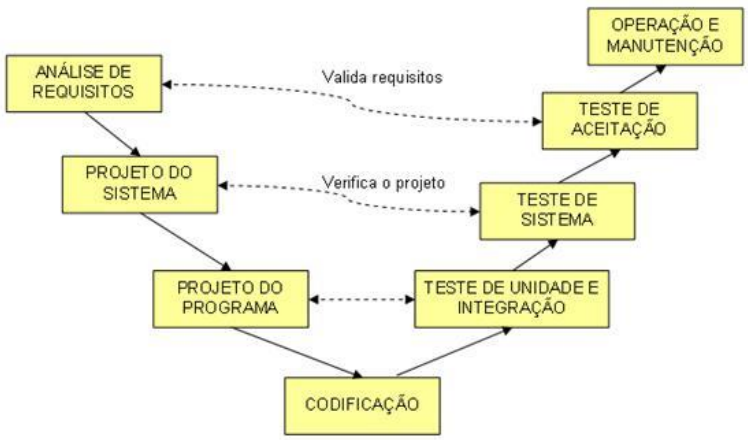
O modelo em cascata tem o grande mérito de ser o primeiro a impor o planejamento e o gerenciamento ao processo de software, que antes era casual. O nome "cascata" foi atribuído em razão da sequência das fases, onde cada fase só começa quando a anterior termina; e da transmissão do resultado da fase anterior como entrada para a fase atual (o fim de cada fase resulta em um documento aprovado). Nesse modelo, portanto, é dada muita ênfase às fases de análise e projeto antes de partir para a programação, a fim de que o objetivo do software esteja bem definido e que sejam evitados retrabalhos, conforme podemos observar na imagem abaixo.



Devido à sua simplicidade, o modelo em cascata é fácil de ser entendido pelo cliente. É um modelo que supõe um início e fim claro e determinado, assim como uma estimativa precisa de custo logo no início, fatores importantes na conquista do cliente.

Modelo em V

Neste modelo, do Ministério de Defesa da Alemanha, 1992, o modelo em cascata é colocado em forma de "V". Do lado esquerdo do V ficam da análise de requisitos até o projeto, a codificação fica no vértice e os testes, desenvolvimento, implantação e manutenção, à direita, como podemos observar na imagem abaixo.



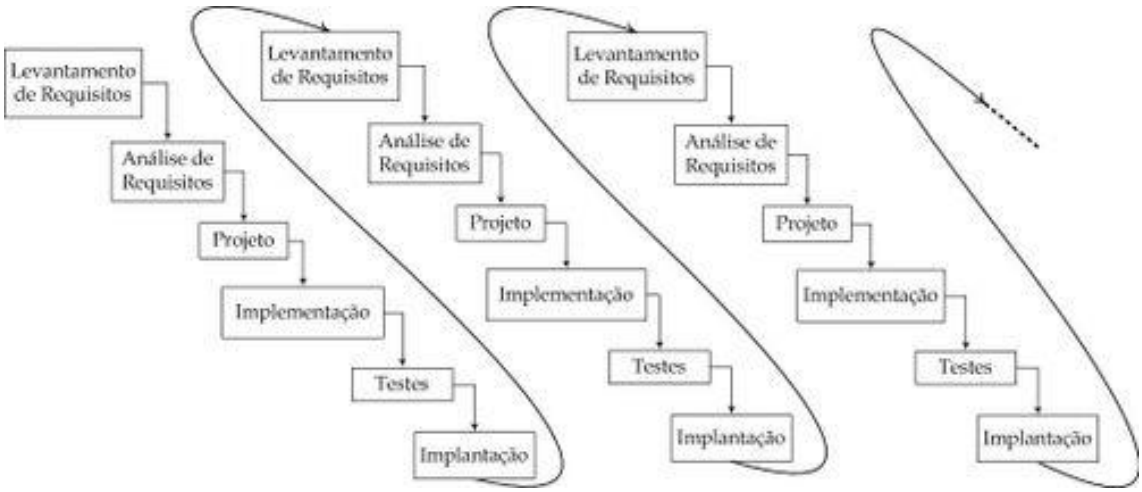
A característica principal desse modelo, que o diferencia do modelo em cascata, é a ênfase dada à verificação e validação: cada fase do lado esquerdo gera um plano de teste a ser executado no lado direito.

Da mesma forma que o modelo em cascata, o cliente só recebe a primeira versão do software no final do ciclo, mas apresenta menos risco, devido ao planejamento prévio dos testes nas fases de análise e projeto.

Modelo Incremental

Neste modelo, de Mills em 1980, os requisitos do cliente são obtidos, e, de acordo com a funcionalidade, são agrupados em módulos. Após este agrupamento, a equipe, junto ao cliente, define a prioridade em que cada módulo será desenvolvido, escolha baseada na importância daquela funcionalidade ao negócio do cliente.

Cada módulo passará por todas as fases "cascata" de projeto, conforme se observa na imagem abaixo, e será entregue ao cliente um software operacional. Assim, o cliente receberá parte do produto final em menos tempo.



Esse ciclo de vida não exige uma equipe muito grande, pois a modularização diminui o escopo de cada incremento, e não há um paralelismo nas atividades. Haverá, por outro lado, uma dificuldade em manter a documentação de cada fase atualizada devido às melhorias no sistema e aos ajustes de requisitos solicitados pelos clientes.

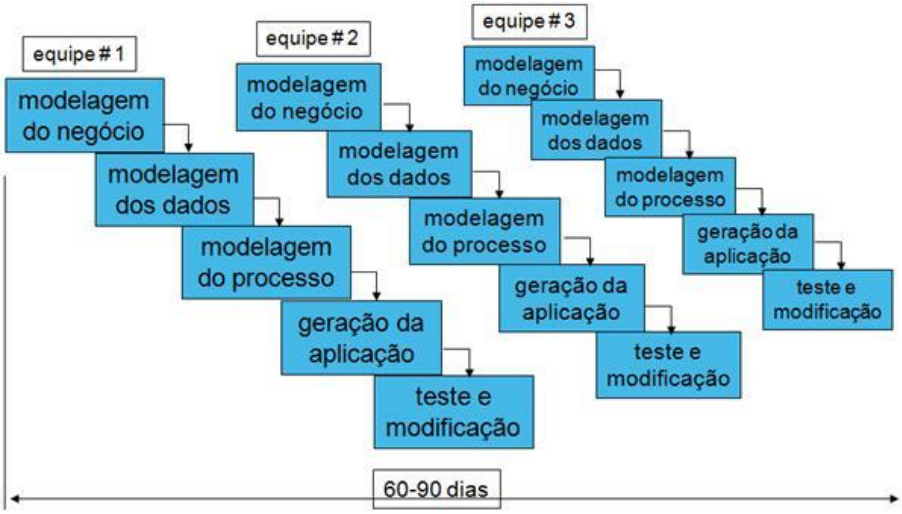
Modelo RAD - Rapid Application Development

Este modelo formalizado por James Martin em 1991, como uma evolução da “prototipagem rápida”, destaca-se pelo desenvolvimento rápido da aplicação. O ciclo de vida é extremamente comprimido, de forma a encontrarem-se exemplos, na literatura, de duração de 60 e 90 dias. É ideal para clientes buscando lançar soluções pioneiras no mercado.

É um ciclo de vida incremental, iterativo, onde é preferível que os requisitos tenham escopo restrito. A diferença principal do ciclo anterior é o forte paralelismo das atividades, requerendo, assim, módulos bastante independentes. Aqui os incrementos são desenvolvidos ao mesmo tempo, por equipes diferentes.

Além do paralelismo, a conquista do baixo tempo se dá graças à compressão da fase de requisitos e da fase de implantação. Isso significa que, na obtenção dos requisitos, costumam-se optar por metodologias mais dinâmicas e rápidas, como workshops ao invés de entrevistas. Permite-se

também um desenvolvimento inicial no nível mais alto de abstração dos requisitos visto o envolvimento maior do usuário e visibilidade mais cedo dos protótipos.



Os sistemas desenvolvidos no ciclo RAD tendem a ter uma padronização de telas muito forte, devido a bibliotecas reutilizáveis e templates, porém tendem a perder em desempenho do sistema e na análise de risco (atividades estas que demandam tempo em qualquer projeto). Assim, é preferível seu uso para softwares de distribuição pequena.

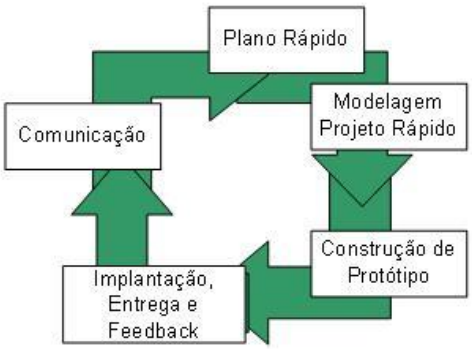
Modelo com Prototipagem

É a construção de um exemplar do que foi entendido dos requisitos capturados do cliente. Pode ser considerado um ciclo de vida ou pode ser usado como ferramenta em outros ciclos de vida.

Um protótipo em engenharia de software pode ser o desenho de uma tela, um software contendo algumas funcionalidades do sistema. São considerados operacionais (quando já podem ser utilizados pelo cliente no ambiente real, ou seja, em produção), ou não operacionais (não estão aptos para serem utilizados em produção). Os protótipos podem ser descartados, ou reaproveitados para evoluírem até a versão final.

No ciclo de vida de prototipagem, não é exigido um conhecimento aprofundado dos requisitos num primeiro momento. Isso é bastante útil quando os requisitos não são totalmente conhecidos, são muitos complexos ou confusos. Desta forma, se o cliente não sabe expressar o que deseja (o que ocorre bastante quando não é um sistema legado), a melhor maneira de evitar que se perca tempo e recursos com uma má interpretação é a construção de modelos, ou seja, de protótipos do que o software faria.

Assim, o cliente experimentará, na prática, como o sistema ou parte dele funcionará. A partir desse primeiro contato, o cliente esclarece o que não foi bem interpretado, aprofunda alguns conceitos e até descobre um pouco mais sobre o que realmente precisa. A partir deste feedback, novos requisitos são colhidos e o projeto ganha maior profundidade. Outro protótipo é gerado e apresentado ao cliente, que retorna com mais feedbacks. Ou seja, o cliente participa ativamente do início ao fim do processo.



É importante relembrar que conforme expliquei anteriormente, não existe um modelo ideal e na maioria dos softwares desenvolvidos são utilizados mais de um modelo de ciclo de vida. Abaixo inseri uma tabela de comparação entre os modelos de ciclo de vida do software apresentados.

Modelo	Foco	Requisitos	Primeira Versão para Cliente	Gerenciamento (escala de 1 a 5 de complexidade)
Cascata	Documento e artefato	Bem conhecido / congelado	Fim do ciclo	1
V	Planejamento de testes	Bem conhecido / congelado	Fim do ciclo	2
Incremental	Incrementos operacionais	Maior abstração / Tratado em módulos	Protótipos operacionais	3
RAD	Rapidez	Escopo restrito / Maior abstração / Tratado em módulos	Protótipos operacionais	4
Prototipagem	Dúvidas nos Requisitos	Abstratos	Protótipos não operacionais	5

PROCESSO DE SOFTWARE

O processo de software é o conjunto de atividades que constituem o desenvolvimento de um sistema computacional. Estas atividades são agrupadas em fases, como: definição de requisitos, análise, projeto, desenvolvimento, teste e implantação. Em cada fase são definidas, além das suas atividades, as funções e responsabilidades de cada membro da equipe, e como produto resultante, os artefatos. O que diferencia um processo de software do outro é a ordem em que as fases vão ocorrer, o tempo e a ênfase dados a cada fase, as atividades presentes, e os produtos entregues.

As atividades do processo de desenvolvimento de software incluem:

- **Análise de requisitos de software**

A extração dos requisitos de um desejado produto de software é a primeira tarefa na sua criação. Embora o cliente, provavelmente, acredite saber o que o software deva fazer, esta tarefa requer habilidade e experiência em Engenharia de Software para reconhecer a incompletude, ambiguidade ou contradição nos requisitos.

- **Especificação**

A especificação é a tarefa de descrever precisamente o software que será escrito, preferencialmente de uma forma matematicamente rigorosa. Na prática, somente especificações mais bem-sucedidas foram escritas para aplicações bem compreendidas e afinadas que já estavam bem desenvolvidas, embora sistemas de software de missão crítica sejam frequentemente bem especificados antes do desenvolvimento da aplicação. Especificações são mais importantes para interfaces externas que devem permanecer estáveis.

- **Arquitetura de Software**

A arquitetura de um sistema de software remete a uma representação abstrata daquele sistema. Arquitetura é concernente à garantia de que o sistema de software irá ao encontro de requisitos do produto, como também assegurar que futuros requisitos possam ser atendidos. A etapa da arquitetura também direciona as interfaces entre os sistemas de software e outros produtos de software, como também com o hardware básico ou com o sistema operacional.

- **Implementação (ou codificação)**

A transformação de um projeto para um código deve ser a parte mais evidente do trabalho da engenharia de software, mas não necessariamente a sua maior porção.

- **Teste**

Teste de partes do software, especialmente onde tenha sido codificado por dois ou mais engenheiros trabalhando juntos, é um papel da engenharia de software.

- **Documentação**

Uma importante tarefa é a documentação do projeto interno do software para propósitos de futuras manutenções e aprimoramentos. As documentações mais importantes são das interfaces externas.

- **Suporte e Treinamento de Software**

Uma grande porcentagem dos projetos de software falha pelo fato de o desenvolvedor não perceber que não importa quanto tempo a equipe de planejamento e desenvolvimento irá gastar na criação do software se ninguém da organização irá usá-lo. As pessoas ocasionalmente resistem à mudança

e evitam aventurar-se em áreas pouco familiares. Então, como parte da fase de desenvolvimento, é muito importante o treinamento para os usuários de software mais entusiasmados, alternando o treinamento entre usuários neutros e usuários favoráveis ao software. Usuários irão ter muitas questões e problemas de software os quais conduzirão para a próxima fase.

- **Manutenção**

A manutenção e melhoria de software lidam com a descoberta de novos problemas e requisitos. Ela pode tomar mais tempo que o gasto no desenvolvimento inicial do mesmo. Não somente pode ser necessário adicionar códigos que combinem com o projeto original, mas determinar como o software trabalhará em algum ponto depois da manutenção estar completa, pode requerer um significativo esforço por parte de um engenheiro de software. Cerca de 2/3 de todos os engenheiros de software trabalham com a manutenção, mas estas estatísticas podem estar enganadas. Uma pequena parte destes trabalha na correção de erros. A maioria das manutenções são para ampliar os sistemas para novas funcionalidades, as quais, de diversas formas, podem ser consideradas um novo trabalho. Analogamente, cerca de 2/3 de todos os engenheiros civis, arquitetos e construtores trabalham com manutenção de uma forma similar.

QUALIDADE DE SOFTWARE

Conceitos

De acordo como SWEBOK 3.0[4], a qualidade de software se refere “as características desejadas de produtos de software, a extensão em que um produto de software em particular possui essas características e aos processos, ferramentas e técnicas que são usadas para garantir essas características”. Portanto, ela está diretamente ligada com a qualidade do processo através do qual o software é desenvolvido, portanto, para se ter qualidade em um produto de software é necessário ter um processo de desenvolvimento bem definido, que deve ser documentado e acompanhado.

Nas normas ISO/EIC 9126-1 e ISO/EIC 25010 a definição de qualidade de software é “a capacidade do produto de software em satisfazer as necessidades implícitas e explícitas quando usado em condições específicas”.

> *“Qualidade de software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais” [Pressman,1994].*

A qualidade de software não pode ser entendida como perfeição. Qualidade é um conceito multidimensional, realizado por um conjunto de atributos, representando vários aspectos

[4] SWEBOK. **Guide to the Software Engineering Body of Knowledge**. Projeto criado pela IEEE para servir como referência em assuntos considerados, de forma geral, como pertinentes a Engenharia de Software. Disponível em: <https://www.computer.org/web/swebok>.

relacionados ao produto: desenvolvimento, manutenção e uso. Qualidade é algo factível, relativo, dinâmico e evolutivo, adequando-se ao nível dos objetivos a serem atingidos (SIMÃO, 2002)[5].

Aspectos Importantes

Diante das definições acima, chegamos a três aspectos importantes. Segundo Crosby[6] "A qualidade é a conformidade aos requisitos". Esta definição nos dá direção sobre qual o caminho seguir para julgar a qualidade de um software.

- Os **requisitos de software** são a base a partir da qual a qualidade é medida. A falta de conformidade aos requisitos significa falta de qualidade.
- Padrões especificados definem um conjunto de **critérios de desenvolvimento** que orientam a maneira segundo a qual o software passa pelo trabalho de engenharia. Se os critérios não forem seguidos, o resultado quase que seguramente será a falta de qualidade.
- Existe um conjunto de **requisitos implícitos** que frequentemente não são mencionados na especificação (por exemplo o desejo de uma boa manutenibilidade).

Note que se o software se adequar aos seus requisitos explícitos, mas deixar de cumprir seus requisitos implícitos, a qualidade do software pode ser comprometida.

Visões de Qualidade

Desenvolvedor

Para o desenvolvedor, a qualidade do software está voltada às características internas. Entretanto, a qualidade interna tem que responder às questões externas que o cliente venha a questionar. Por exemplo:

- As funções X estão disponíveis e são executadas eficientemente?
- Funciona corretamente em imprevistos, como, por exemplo, efetuar débito em uma conta com saldo negativo?
- O software é seguro, ou seja, evita que pessoas ou sistemas não autorizados tenham acesso aos dados para modificar?
- É fácil de usar ou requer muito treinamento?
- É fácil de integrar com outros sistemas existentes?
- Aceita trabalhar com arquivos de outros sistemas ou enviar dados para outros sistemas?

---

[5] SIMÃO, R. P. S. **Características de Qualidade para Componentes de Software**. 2002. Dissertação (Mestrado em Informática Aplicada) - Programa de Pós-Graduação em Informática, Fundação Educacional Edson de Queiroz, Universidade de Fortaleza, Fortaleza.

[6] Philip Bayard Crosby, uma das mais importantes personalidades da qualidade.

Por essas questões, os desenvolvedores não podem menosprezar o papel do usuário/cliente, não podendo se esquecer das necessidades implícitas do uso.

Usuário

Na visão do usuário, seu interesse está no uso do software, na sua funcionalidade, no desempenho e nos efeitos que o uso possa produzir na sua empresa ou organização. Ou seja, o cliente valoriza a qualidade do software de acordo com sua resposta às suas necessidades.

Gerente

Existe ainda a visão do gerente da empresa, onde a qualidade do produto não pode ser desvinculada dos interesses da organização. Os gerentes avaliam aspectos de conformidade em relação aos requisitos dos clientes e desenvolvedores e também aspectos de custo e cronograma.

MODELOS DE QUALIDADE

A avaliação da qualidade de produtos de software normalmente é feita através de modelos de avaliação de qualidade. Esses modelos descrevem e organizam as propriedades de qualidade do produto em avaliação. Os modelos de avaliação mais aceitos e usados no mercado são:

- **Norma ISO/IEC 9126**, proposta pela ISO (*International Organization for Standardization*).
- **CMMI** (*Capability Maturity Model Integration*), proposto pelo CMM (*Capability Maturity Model*);

Entretanto, nosso edital também cobra o modelo de processo de software **MPS.BR** (Melhoria de Processos do Software Brasileiro), criado em 2003 pela Softex (Associação para Promoção da Excelência do Software Brasileiro).

Norma ISO/IEC 9126

O modelo proposto pela ISO/IEC 9126 (NBR 13596) tem como objetivo servir de referência básica na avaliação de produto de software.

A ISO/IEC em sua primeira parte, descreve um modelo de qualidade do produto de software, composto por duas partes: **qualidade interna e externa e qualidade de uso**, como pode ser observado na tabela abaixo.

Tipo de Qualidade	Descrição
Qualidade Interna e Externa	Especifica seis características para qualidade interna e externa, as quais por sua vez são subdivididas em sub características que são manifestadas

	externamente, quando o software é utilizado como parte de um sistema computacional, e são resultantes de atributos internos do software.
Qualidade de uso	Especifica quatro características de qualidade em uso, mas não apresenta o modelo de qualidade em uso além do nível de característica, que é para o usuário, o efeito combinado das seis características de qualidade do produto de software.

Na tabela abaixo temos o modelo de propósito geral da qualidade interna e externa que define as seis amplas categorias de características de qualidade de software. Estas características estão relacionadas diretamente com os requisitos não funcionais esperados de um software.

Características	Sub Características	Significado
Funcionalidade (como as funções e propriedades específicas do produto, satisfazem as necessidades do usuário)	**Adequação**	Existência de um conjunto de funções apropriadas para as tarefas requeridas;
	Acurácia	Produção de resultados ou efeitos corretos;
	Interoperabilidade	Habilidade de interação do produto de software com outros produtos;
	Segurança de acesso	Aptidão para evitar acessos não autorizados a programas e dados;
	Conformidade	O produto está de acordo com as convenções, as normas ou os regulamentos estabelecidos.
Confiabilidade (como o produto de software é capaz de manter seu nível de desempenho, ao longo do tempo, nas condições estabelecidas)	**Maturidade**	Estado de maturação do software, detectada por sua baixa frequência de falhas;
	Tolerância a falhas	O nível de desempenho é mantido, quando ocorrem falhas;
	Recuperabilidade	Existem mecanismos que restabelecem e restauram os dados após a ocorrência de falhas.

Usabilidade (o esforço necessário para a utilização do sistema, baseado em um conjunto de implicações e de condições do usuário)	**Inteligibilidade**	Facilidade de entendimento dos conceitos utilizados no produto de software;
	Apreensibilidade	Facilidade de aprendizado do software;
	Operacionalidade	Faculdade de operar e controlar operações pertinentes ao software.
Eficiência (como os recursos e os tempos envolvidos são compatíveis com o nível de desempenho requerido pelo software)	**Comportamento em relação aos recursos**	Relaciona-se com a quantidade dos recursos empregados.
	Comportamento em relação ao tempo	Refere-se ao tempo de resposta de processamento.
Manutenibilidade (refere-se ao esforço necessário para a realização de alterações específicas, no produto de software)	**Analisibilidade**	Característica de ser possível diagnosticar deficiências e causas de falhas;
	Modificabilidade	Característica que o produto deve ter de forma a facilitar modificações e remoções de defeitos;
	Estabilidade	Ausência de riscos ou ocorrências de defeitos inesperados no software;
	Testabilidade	Facilidade de o produto ser testado.
Portabilidade (facilidade do software poder ser transferido de um ambiente para outro)	**Adaptabilidade**	Faculdade de o produto poder ser adaptado a novos ambientes;
	Capacidade para ser instalado	Facilidade de instalação do produto de Software Conformidade com padrões;
	Capacidade para substituir	O produto de software pode ser substituído por outro, sem grandes esforços.
	Conformidade	O produto está segundo os padrões ou convenções de portatilidade.

A tabela abaixo apresenta as quatro características definidas na qualidade de uso.

Características
Eficácia: capacidade do produto de software de permitir que seus usuários atinjam metas especificadas com acurácia e completitude, em um contexto de uso especificado.
Produtividade: capacidade do produto de software de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida, em um contexto de uso especificado.
Segurança: capacidade do produto de software de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedades ou ao ambiente, em um contexto de uso especificado.
Satisfação: capacidade do produto de satisfazer usuários, em um contexto de uso especificado.

CMMI (versão 1.3)

O CMMI, sigla para *Capability Maturity Model Integration* (Modelo de Maturidade em Capacitação Integração) é um modelo de maturidade que sugere o que deve ser aplicado pelas organizações nos processos de software. Ele é baseado nas melhores práticas para desenvolvimento e manutenção de produtos, com ênfase tanto em engenharia de sistemas quanto em engenharia de software.

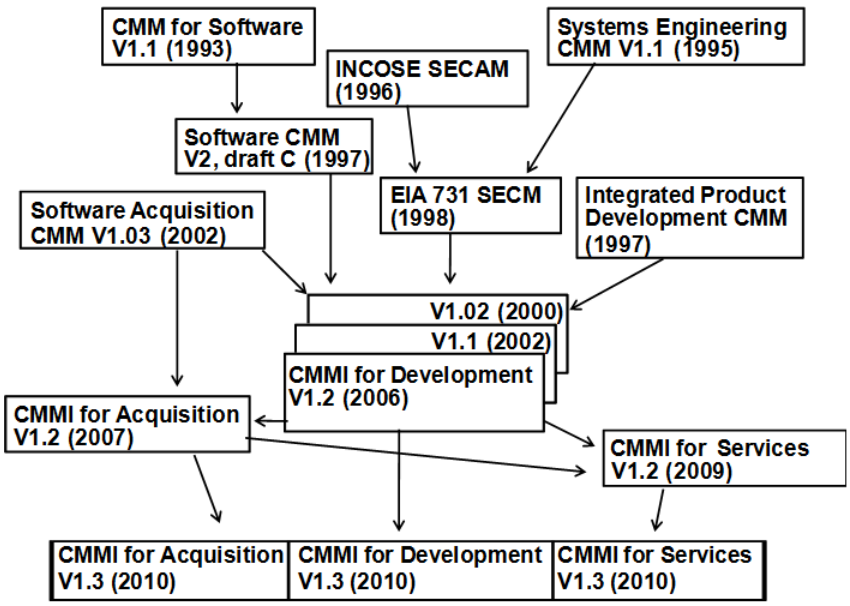
A versão mais atual do CMMI é a 1.3 publicada em 27 de outubro de 2010 e ela possui os seguintes modelos para três áreas de interesse:

- **CMMI for Development** (CMMI-DEV), voltado ao processo de desenvolvimento de produtos e serviços. Contém práticas que cobrem: Gerenciamento de projetos, Gerenciamento de processos, Engenharia de sistemas, Engenharia de hardware, Engenharia de software e outros processos de suporte usados em desenvolvimento e manutenção.
- **CMMI for Acquisition** (CMMI-ACQ), voltado aos processos de aquisição e terceirização de bens e serviços; e
- **CMMI for Services** (CMMI-SVC), voltado aos processos de empresas prestadoras de serviços.

Evolução do CMMI

Antes de receber o nome CMM, o primeiro framework *Capability Maturity Model* foi publicado em 1989 por Watts Humphrey no livro *Managing the Software Process*. Alguns anos antes, o DoD tinha feito um pedido de propostas para responder o motivo do gasto excessivo de dinheiro em softwares que nunca eram entregues ou quando eram nunca atendiam as funcionalidades solicitadas.

No espaço de poucos anos, o CMM deu origem a vários outros modelos, desenvolvidos para responder às necessidades de projetos que não tinham nada a ver com o desenvolvimento de software. O CMM e as suas variantes também passaram a ser cada vez mais utilizados a nível internacional e pela indústria comercial. Como as organizações que tentavam adotar mais de um modelo num dado projeto, elas pediram ao SEI (*Software Engineering Institute* - responsável pelos CMMs) que consolidasse os vários CMMs num só modelo. Foi assim que surgiu o CMMI em 1998, criado por uma equipa constituída por representantes da indústria, governo e SEI.



Representações

O CMMI possui duas representações: "contínua" ou "por estágios". Estas representações permitem a organização utilizar diferentes caminhos para a melhoria de acordo com seu interesse.

- Representação Continua

Possibilita a organização utilizar a ordem de melhoria que melhor atender os objetivos de negócio da empresa. É caracterizado por Níveis de Capacidade (*Capability Levels*).

Nesta representação a capacidade é medida por processos separadamente, onde é possível ter um processo com nível um e outro processo com nível cinco, variando de acordo com os interesses da empresa.

A representação contínua é indicada quando a empresa deseja tornar apenas alguns processos mais maduros, quando já utiliza algum modelo de maturidade contínua ou quando não pretende usar a maturidade alcançada como modelo de comparação com outras empresas.

- Representação Por Estágios

Disponibiliza uma sequência pré-determinada para melhoria baseada em estágios que não deve ser desconsiderada, pois cada estágio serve de base para o próximo. É caracterizado por Níveis de Maturidade (*Maturity Levels*).

Nesta representação a maturidade é medida por um conjunto de processos. Assim é necessário que todos os processos atinjam nível de maturidade dois para que a empresa seja certificada com nível dois. Se quase todos os processos forem nível três, mas apenas um deles estiver no nível dois a empresa não irá conseguir obter o nível de maturidade três.

Esta representação é indicada quando a empresa já utiliza algum modelo de maturidade por estágios, quando deseja utilizar o nível de maturidade alcançado para comparação com outras empresas ou quando pretende usar o nível de conhecimento obtido por outros para sua área de atuação.

Nível de maturidade

Por meio de um processo formal de avaliação, uma organização é classificada em um "nível de maturidade" que varia de um a cinco.

O nível de maturidade indica em que medida os processos daquela organização são maduros. Quanto maior o nível de maturidade, melhores e mais maduros são os processos.

Os níveis são organizados de modo a estabelecer as prioridades na condução de programas de melhoria do processo de software. Cada nível é considerado como pré-requisito para o nível seguinte, que se apoia nas competências que a organização desenvolveu no nível imediatamente inferior.

Abaixo temos uma imagem mostrando como os níveis de maturidade são divididos.

As principais características dos níveis acima são:

Nível 1 - Inicial. Imaturidade organizacional; os processos são improvisados e geralmente não são seguidos; compromissos de prazo e custo não são cumpridos; o planejamento não é feito com base em estimativas; as qualidades, procedimentos e conhecimentos pertencem às pessoas e não aos projetos; as chances de sucesso dependem das habilidades pessoais dos gerentes e desenvolvedores

Nível 2 - Gerenciado. Políticas e procedimentos para gerenciar o desenvolvimento de software estão definidas e são obedecidas; o planejamento é baseado em estimativas e na experiência anterior de outros projetos; os projetos utilizam processos definidos, usados, disseminados, documentados, medidos e fiscalizados com rotinas de melhoria; os processos afetados são puramente gerenciais (não técnicos) e pertencem aos projetos e não às pessoas.

Nível 3 - Definido. Os processos utilizados são estabelecidos e padronizados em toda a organização; processos técnicos passam a ser considerados ao lado dos processos gerenciais; tanto os processos gerenciais quanto os técnicos passam a ser repetidos; os processos pertencem a organização e não mais aos projetos.

Nível 4 - Quantitativamente Gerenciado. São estabelecidas metas quantitativas para os processos e produtos, medidas de qualidade e produtividade são coletadas em todos os projetos; é estabelecido controle estatístico de processos; a gestão passa a ser feitas com bases quantitativas.

Nível 5 - Otimização. A organização está engajada na melhoria contínua de seus processos; identificação de pontos fracos e defeitos; ações preventivas sobre causas; mudanças mais significativas de processos e/ou tecnologias são feitas a partir de análise de custo/benefício com base em dados quantitativos.

O modelo CMMI possui ao todo 22 áreas de processo para que a organização alcance o nível máximo de maturidade. Abaixo temos uma lista com as áreas correspondentes a cada nível.

- Nível 1: Inicial
  - Não possui áreas de processo.

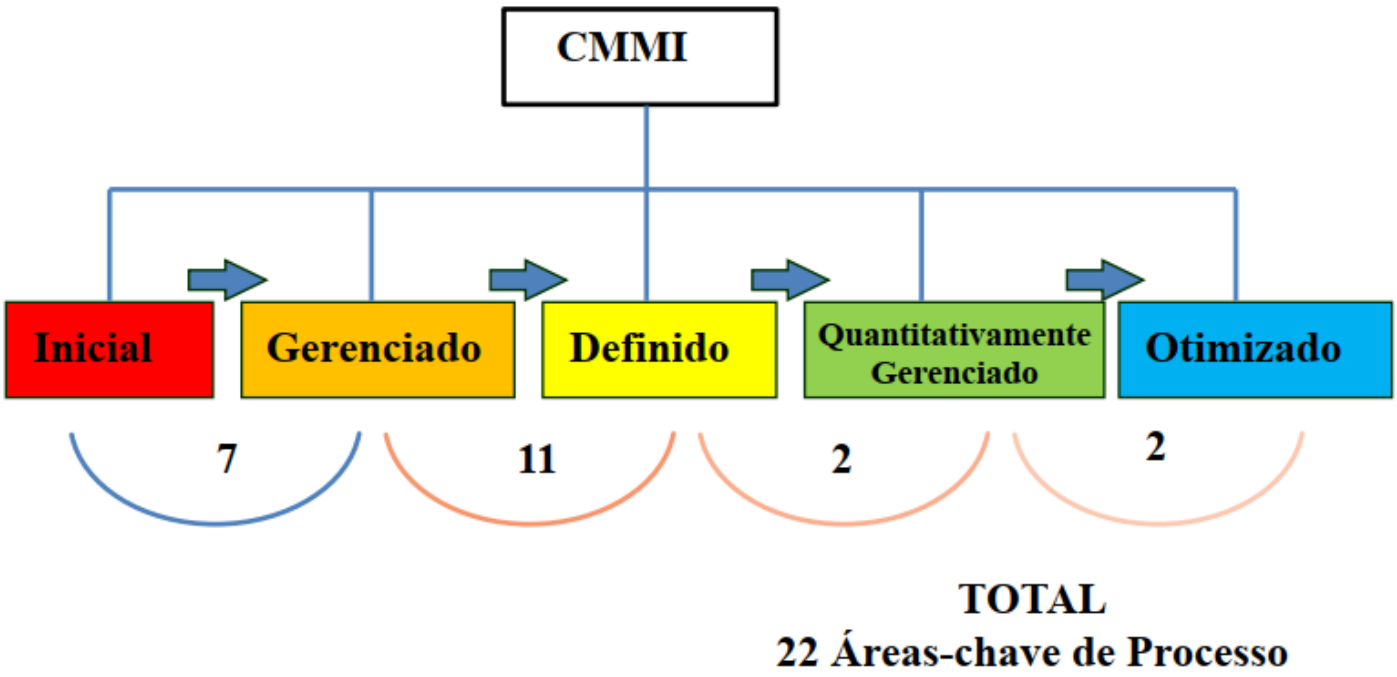
- Nível 2: Gerenciado
  - Gerenciamento de Requisitos - REQM (Requirements Management)
  - Planejamento de Projeto - PP (Project Planning)
  - Acompanhamento e Controle de Projeto - PMC (Project Monitoring and Control)
  - Gerenciamento de Acordo com Fornecedor - SAM (Supplier Agreement Management)
  - Medição e Análise - MA (Measurement and Analysis)
  - Garantia da Qualidade de Processo e Produto - PPQA (Process and Product Quality Assurance)
  - Gerência de Configuração - CM (Configuration Management)

- Nível 3: Definido:
  - Desenvolvimento de Requisitos - RD (Requirements Development)

- Solução Técnica - TS (Technical Solution)
- Integração de Produto - PI (Product Integration)
- Verificação - VER (Verification)
- Validação - VAL (Validation)
- Foco de Processo Organizacional - OPF (Organizational Process Focus)
- Definição de Processo Organizacional - OPD (Organizational Process Definition)
- Treinamento Organizacional - OT (Organizational Training)
- Gerenciamento Integrado de Projeto - IPM (Integrated Project Management)
- Gerenciamento de Riscos - RSKM (Risk Management)
- Análise de Decisão e Resolução - DAR (Decision Analysis and Resolution)

- Nível 4: Quantitativamente gerenciado
  - Desempenho de Processo Organizacional - OPP (Organizational Process Performance)
  - Gerenciamento Quantitativo de Projeto - QPM (Quantitative Project Management)

- Nível 5: Otimização
  - Gestão do Desempenho Organizacional - OPM (Organizational Performance Management)
  - Análise Causal e Resolução - CAR (Causal Analysis and Resolution)



Nível de capacidade

Um nível de capacidade para uma área de processo é atingido quando todos os objetivos genéricos são satisfeitos até aquele nível.

Os quatro níveis de capacidade, cada um servindo de fundação para melhoria contínua de processos, são designados pelos números de 0 a 3:

Nível 0 – Incompleto. Neste nível de capacidade, a Área de Processo não alcança uma ou mais metas específicas. Como já dissemos, no CMMI a coisa é binária. Não adianta fazer o quase. É tudo ou nada. Assim sendo, a Área de Negócio classificada no nível zero significa a mesma não é executada ou é executada de forma parcial.

Nível 1 – Executado. Neste nível, a Área de Processo satisfaz todas as suas metas específicas, o que garante que o trabalho necessário é feito a fim de transformar entradas bem definidas em saídas adequadas para aquela Área de Processo.

Nível 2 – Gerenciado. Neste nível, a Área de Processo é planejada e executada de acordo com uma política e há o emprego adequado de recursos e pessoas. Além disso, a Área de Processo produz resultados controlados uma vez que é monitorado e revisado. Cabe aqui uma observação importante, apesar da Área de Processo ser executada de acordo com uma política, não significa dizer que seja algo padronizado, pois esta é uma característica específica do próximo nível. Se pudermos exprimir em palavras a diferença entre Política e Padrão diria que este (padrão) envolve ferramentas, procedimentos, métodos de trabalho definidos para os projetos da organização, enquanto aquele (Política) representa princípios gerais, diretrizes a serem seguidas pela Área de Processo.

Nível 3 – Definido. Neste nível, a Área de Processo, além de fazer tudo dos níveis anteriores, é uma adaptação do processo padrão da organização. A Área de Processo fornece informações para sua melhoria. Diferentemente do que se possa pensar, é neste nível de capacidade que uma Área de Processo começa a fazer uma melhoria em seu próprio processo. No nível 3, a melhoria está baseada na identificação de pontos fortes e pontos fracos da Área de Processo.

Abaixo temos uma tabela, atualizada de acordo com a versão 1.3 do CMMI, com os níveis de capacidade X níveis de maturidade.

Nível	Capacidade	Maturidade
0	Incompleto	
1	Realizado	Inicial
2	Gerenciado	Gerenciado
3	Definido	Definido
4		Quantitativamente gerenciado
5		Em otimização

MPS.BR

O modelo MPS.BR foi criado em 2003 pela Softex (Associação para Promoção da Excelência do Software Brasileiro), com o objetivo de melhorar a capacidade de desenvolvimento de softwares por empresas brasileiras.

O MPS.BR funciona como um selo que indica o nível de maturidade da empresa em relação às práticas relacionadas ao desenvolvimento de software. Esse selo funciona como uma garantia que a empresa utiliza as boas práticas e possui condições de desenvolver um software de qualidade, dentro dos custos e prazos estabelecidos. Algumas licitações estão exigindo esse selo para as empresas poderem participar.

Entre as metas do MPS.BR temos a que visa definir e aprimorar um modelo, reconhecido nacional e internacionalmente, de melhoria e avaliação de processo de software, visando preferencialmente as micro, pequenas e médias empresas, com o objetivo de atender as necessidades do negócio. Também são estabelecidos um processo e um método de avaliação, que dá sustentação e garante que o MPS.BR está sendo empregado de forma coerente às suas definições.

A base técnica para a construção e aprimoramento do modelo MPS.BR é composta pelas normas e modelos internacionalmente reconhecidos: o CMMI; a NBR ISO/IEC 12207 – Processo de Ciclo de Vida de Software; as emendas 1 e 2 da norma internacional ISO/IEC 12207; e a ISO/IEC 15504 – Avaliação de Processo.

O programa mobilizador MPS.BR está dividido em 3 componentes:

- Modelo de Referência (MR-MPS);
- Método de Avaliação (MA-MPS);
- Modelo de Negócio (MN-MPS).

MR-MPS

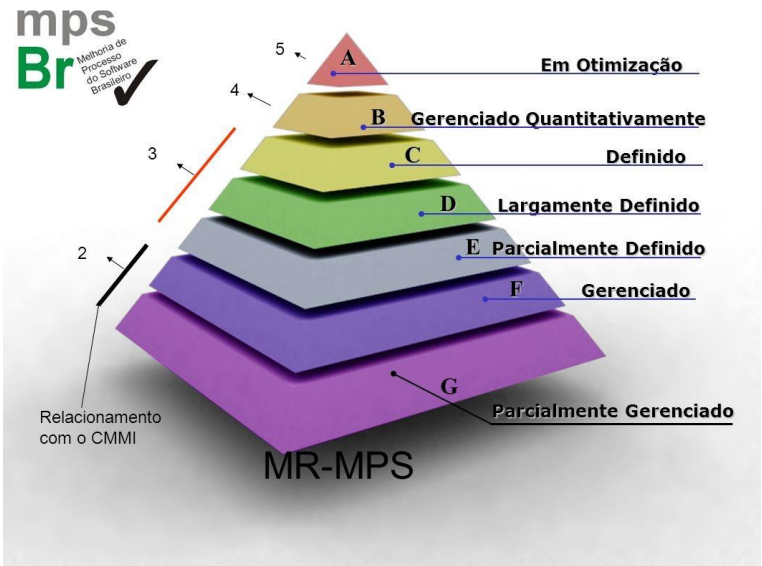
O Modelo de Referência (MR-MPS) define níveis de maturidade que são uma combinação entre os processos e sua capacidade. Isso permite avaliar e atribuir graus de efetividade na execução dos processos. As atividades e tarefas necessárias para atender ao propósito e aos resultados.

- Níveis de maturidade

Os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria da implementação de processos na organização. O nível de maturidade em que se encontra uma organização permite prever o seu desempenho futuro ao executar um ou mais processos. O MR-MPS define sete níveis de maturidade:

1) A (Em Otimização);
2) B (Gerenciado Quantitativamente);
3) C (Definido);
4) D (Largamente Definido);
5) E (Parcialmente Definido);
6) F (Gerenciado);
7) G (Parcialmente Gerenciado).

Na imagem abaixo podemos observar que esses níveis são representados como uma pirâmide e formam uma escala de maturidade.



A escala de maturidade se inicia no nível G e progride até o nível A. Para cada um destes sete níveis de maturidade é atribuído um perfil de processos que indicam onde a organização deve colocar o esforço de melhoria. O progresso e o alcance de um determinado nível de maturidade MPS se obtêm, quando são atendidos os propósitos e todos os resultados esperados dos respectivos processos e dos atributos de processo estabelecidos para aquele nível.

A divisão em estágios, embora baseada nos níveis de maturidade do CMMISE/SWSM tem uma graduação diferente, com o objetivo de possibilitar uma implementação e avaliação mais adequada

as micro, pequenas e médias empresas. A possibilidade de se realizar avaliações considerando mais níveis também permite uma visibilidade dos resultados de melhoria de processos em prazos mais curtos.

- Processos

Cada nível de maturidade possui suas áreas de processo, onde são analisados os processos fundamentais (aquisição, gerência de requisitos, desenvolvimento de requisitos, solução técnica, integração do produto, instalação do produto, liberação do produto), processos organizacionais (gerência de projeto, adaptação do processo para gerência de projeto, análise de decisão e resolução, gerência de riscos, avaliação e melhoria do processo organizacional, definição do processo organizacional, desempenho do processo organizacional, gerência quantitativa do projeto, análise e resolução de causas, inovação e implantação na organização) e os processos de apoio (garantia de qualidade, gerência de configuração, validação, medição, verificação, treinamento).

- Capacidade do Processo

A capacidade do processo é representada por um conjunto de atributos de processo descrito em termos de resultados esperados. A capacidade do processo expressa o grau de refinamento e institucionalização com que o processo é executado na organização. No MPS, à medida que a organização evolui nos níveis de maturidade, um maior nível de capacidade para desempenhar o processo deve ser atingido pela organização.

O atendimento aos atributos do processo (AP), através do atendimento aos resultados esperados dos atributos do processo (RAP) é requerido para todos os processos no nível correspondente ao nível de maturidade, embora eles não sejam detalhados dentro de cada processo. Os níveis são acumulativos, ou seja, se a organização está no nível F, esta possui o nível de capacidade do nível F que inclui os atributos de processo dos níveis G e F para todos os processos relacionados no nível de maturidade F (que também inclui os processos de nível G).

A capacidade do processo no MPS possui os seguintes atributos de processos (AP):

1) **AP 1.1 - O processo é executado** (Nível G) - o processo realiza o que foi proposto para ele;
2) **AP 2.1 - O processo é gerenciado** (Nível G e F) - a execução do processo é gerenciada;
3) **AP 2.2 - Os produtos de trabalho do processo são gerenciados** (Nível G e F) - os produtos de trabalho realizados pelo processo são gerenciados de forma adequada;
4) **AP 3.1 - O processo é definido** (Níveis E, D e C) - existe um padrão e que o mesmo apoia a implementação do processo;
5) **AP 3.2 - O processo está implementado** (Níveis E, D e C) - o processo, padronizado, é de fato implementado para atingir os seus objetivos;
6) **AP 4.1 – O processo é medido** (Nível B) - algumas medições são usadas para assegurar que o desempenho do processo ajuda a alcançar os objetivos para o qual este processo foi proposto;
7) **AP 4.2 – O processo é controlado** (Nível B) - o processo é controlado estatisticamente, permitindo se ter previsibilidade, estabilidade e capacidade de execução;
8) **AP 5.1 – O processo é o objeto de inovações** (Nível A) - as mudanças no processo são identificadas a partir da análise dos seus indicadores e da investigação de possíveis inovações;

9) **AP 5.2 – O processo é otimizado continuamente** (Nível A) - as mudanças no processo têm, de fato, impacto no alcance dos objetivos, no que tange os aspectos relevantes de melhoria do mesmo.

Note que ao lado de cada atributo relacionei o nível de maturidade ao qual ele está interligado.

Também é importante que você saiba que os atributos de processos 4.1, 4.2, 5.1 e 5.2 são exclusivos da versão 1.2.

MA-MPS

O Método de Avaliação é composto pelo processo de avaliação MPS, método de avaliação MPS e características da qualificação dos avaliadores. Este método permite a realização de avaliações segundo o Modelo MPS.

O MA-MPS foi definido em conformidade com os requisitos para modelos de referência de processo e métodos de avaliação de processos estabelecidos na norma ISO/IEC 15504-2 e atende aos requisitos específicos do Programa MPS.BR. Desta forma, o método está em conformidade com a ISO/IEC 15504 e é compatível com o método SCAMPI [1] para avaliação segundo o modelo CMMI, também definido com base na ISO/IEC 15504.

O processo de avaliação é composto por quatro sub processos:

1) Contratar a avaliação;
2) Preparar a realização da avaliação;
3) Realizar a avaliação;
4) Documentar os resultados da avaliação.

MN-MPS

O Modelo de Negócio MN-MPS descreve regras de negócio para: implementação dos Modelos de Referencias (MR-MPS) pelas instituições implementadoras; avaliação seguindo o Método de Avaliação (MA-MPS) pelas instituições avaliadoras; habilitação de Consultores de Aquisição (CA) de software e serviços correlatos e credenciamento de Instituições de Consultoria de Aquisição (ICA); realização tanto de cursos e provas oficiais quanto de cursos especiais e workshops anuais do MPS, para capacitação de pessoas no Modelo MPS.

O MN-MPS possui três domínios:

1) na retaguarda, domínio do Programa MPS.BR, coordenado pela Softex;
2) na linha de frente, domínio tanto das Instituições Implementadoras (II) e Instituições Avaliadoras (IA) do MPS.BR quanto das Instituições de Consultoria de Aquisição (ICA) e instrutores dos cursos MPS;
3) no lado dos clientes, domínio das empresas que adotaram o Modelo MPS e Instituições Organizadoras de Grupos de Empresas (IOGE), compreendendo:

- o Modelo de Negócio Cooperado (MNC) – pacote para grupos de empresas, próprio para Micro, Pequenas e Médias Empresas (MPME) que desejam compartilhar custos e esforços principalmente na implementação do Modelo MPS visando uma avaliação seguindo o MA-MPS;
- o Modelo de Negócio Específico (MNE) – personalizado para uma empresa, próprio para organizações que desejam exclusividade na implementação e avaliação MPS.

Processo de Garantia da Qualidade - GQA

O propósito do processo de garantia da qualidade é assegurar que os produtos de trabalho e a execução dos processos estejam em conformidade com os planos, procedimentos e padrões estabelecidos. (MPS.BR – Guia Geral, Setembro 2009, p.31)

Este processo está dividido em outros 4 procedimentos a serem executados, que são descritos abaixo cada, em paralelo foi sugerido através da metodologia ágil Scrum como podemos implementar tais processos.

GQA 1. A aderência dos produtos de trabalho aos padrões, procedimentos e requisitos aplicáveis é avaliada objetivamente, antes dos produtos serem entregues e em marcos predefinidos ao longo do ciclo de vida do projeto; Ideias sobre evidências: Relatórios de auditorias de GQA executadas, segundo plano de GQA em pontos do ciclo de vida do projeto. Por exemplo, auditoria de GQA no final do Pregame, durante os sprints do Development (presprint, sprint e postsprint) e durante o postgame . Os relatórios conterão resultados das auditorias de produtos de trabalho, realizados através de check-lists.

GQA 2. A aderência dos processos executados às descrições de processo, padrões e procedimentos é avaliada objetivamente; Ideias sobre evidências: Relatórios de auditoria de GQA executado, segundo plano de GQA em pontos do ciclo de vida do projeto. Por exemplo, auditoria de GQA no final do Pregame, durante os sprints do Development (presprint, sprint e postsprint) e durante o postgame. Os relatórios conterão resultados de auditoria de processos, realizados através de check-lists. A mesma instância de auditoria avaliará tanto produto quanto processo.

GQA 3. Os problemas e as não-conformidades são identificados, registrados e comunicados; Ideias sobre evidências: Os problemas detectados durante a auditoria (NC-Não conformidades), serão registrados numa ferramenta de issue-tracker, por exemplo, que deverá permitir o acompanhamento até a sua resolução. Um dos pontos que poderá ser adotado no Scrum, o que potencializará a conscientização sobre erros e, principalmente, como evitá-los, será a discussão dessas NC dentro das reuniões da equipe (daily scrum ou sprint review), por exemplo.

GQA 4. Ações corretivas para as não-conformidades são estabelecidas e acompanhadas até as suas efetivas conclusões. Quando necessário, o escalamento das ações corretivas para níveis superiores é realizado, de forma a garantir sua solução; Ideias sobre evidências: O foco aqui é o acompanhamento da NC até a sua conclusão e a possibilidade de escalonamento, em caso de impasses na resolução. Esse ponto tem um componente cultural relativo ao fato dos colaboradores

serem “observados” por alguém de fora do “team” e auditado nos seus erros. A filosofia dos métodos ágeis, por conterem traços de comportamento mais libertos e de auto-gestão, talvez até facilite esse aspecto.

UML – UNIFIED MODELING LANGUAGE

A necessidade de um técnica-padrão que visa cobrir todas as etapas de desenvolvimento de um sistema de informação. O espectro que vai da análise de requisitos, passando pela modelagem, projeto, implementação até chegar a implantação. Um destas técnicas é conhecida como UML – *Unified Modeling Laguage*. Ela oferece um mecanismo na forma de notação diagramática e sintaxe de linguagem associada para cobrir todo o ciclo de vida.

A UML combina conceitos comumente aceitos de muitos métodos e metodologias orientados a objetos (O-O). Ela é genérica independente de linguagem e plataforma. A UML tem muitos tipos diagramas que podem ser divididos em duas categorias:

> **Diagramas estruturais**. Estes descrevem os relacionamentos estruturais ou estáticos entre objetos de esquema, objetos de dados componentes de software. Incluem os diagramas de classes, de objetos, de componentes e de implementação.
> **Diagramas comportamentais**. Sua finalidade é descrever o comportamento ou relacionamento dinâmico entre componentes. Incluem diagramas de caso de uso, de seq uência, de colaboração, de estados e de atividades.

Pense que um sistema tem aspectos estáticos e dinâmicos, esses dois tipos de diagrama descrevem essas características. Vamos dar um exemplo de cada um dos tipos de diagrama apenas para ajudar você na fixação do conteúdo. O **Diagrama de Classes** oferece um ótimo exemplo do tipo de diagrama estrutural e fornece um conjunto inicial de elementos de notação que todos os outros diagramas de estrutura usam. O propósito do diagrama de classes é mostrar os tipos que estão sendo modelados no sistema.

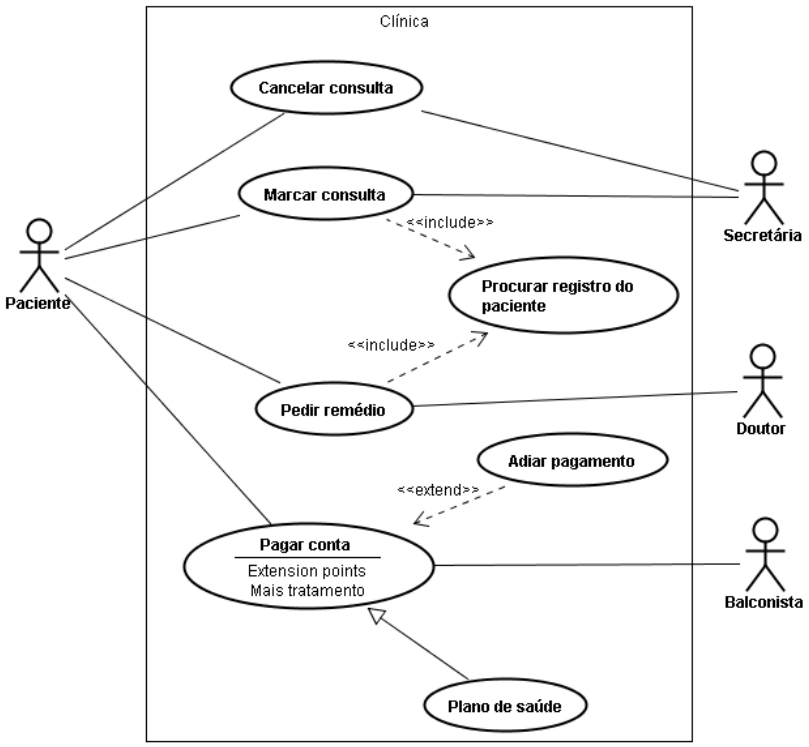
Uma classe é representada na forma de um retângulo, contendo duas linhas que separam 3 partes. A primeira contém no nome da classe, a segunda os atributos da classe e a última os métodos da mesma. Vejamos uma figura que ajude a entender melhor esses conceitos:

Concurseiro
nome: String concursoAlvo: String identidade: Integer dataDeNascimento: Date
getConcursoAlvo() : String getIdade(): Integer

Observe a classe acima, temos a classe Concurseiro com os atributos nome, concursoAlvo, indentidade e dataDeNascimento. Veja que cada atributo tem um tipo de dado associado. Por fim, o retângulo mostra as operações que podem ser executadas com os dados desta classe, conhecidos como métodos. Um detalhamento maior sobre o que seria um diagrama de classe pode ser visto neste artigo[7].

O diagrama comportamental que usaremos como exemplo é o diagrama de Casos de Uso. O diagrama de casos de uso tem o objetivo de **auxiliar a comunicação entre os analistas e o cliente**. Ele descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário. O cliente deve ver no diagrama de casos de uso as principais funcionalidades de seu sistema.

O diagrama de Caso de Uso é representado por atores, casos de uso e relacionamentos entre estes elementos. Um **ator** é representado por um boneco e um rótulo com o nome do ator. Um ator é um usuário do sistema, que pode ser um usuário humano ou um outro sistema computacional. Um **caso de uso** é representado por uma elipse e um rótulo com o nome do caso de uso. Um caso de uso define uma grande função do sistema. Os relacionamentos ajudam a descrever casos de uso. A figura abaixo descreve um diagrama de caso de uso. Nela temos os atores Paciente, Secretária, Doutor e Balconista que fazem acesso a diferentes macro funcionalidades do sistema, ou seja, os casos de uso.



Antes de você pergunte o que são os termos <<include>> e <<extends>> na figura acima, deixa eu tentar explicar de uma forma bem simples. O <<include>> afirma que um caso de uso depende do outro, tente observar na figura, para marcar uma consulta é necessário ter os dados do paciente. Já

[7] https://www.ibm.com/developerworks/br/rational/library/content/RationalEdge/sep04/bell/index.html

o <<extends>> incluí uma funcionalidade extra que não é obrigatória, por exemplo, adiar pagamento é uma possibilidade na ação de pagar conta.

Forte abraço e bons estudos.

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.